Aspera Orchestrator - User Guide

Version 2.4.1

[Aspera Software Services Team]

ABSTRACT

[This User Guide describes how to use the available features of Aspera Orchestrator 2.4.1. The Guide also gives examples of the API. Various components such as Workflows, Work Orders and other Orchestrator runtime elements respond to these API requests.]

CONTENTS

INTRODUCTION

The Orchestrator Admin guide provides documentation and usage procedure about Orchestrator features. The version 2.4 of Admin guide contains information about features released with Orchestrator version 2.4

Please contact the Aspera Orchestrator team or Aspera Support (support@asperasoft.com) for any further questions and/or suggestions.

ORCHESTRATOR INSTALLATION DIRECTORIES

Operating Systems: RHEL 5.x, 6.x or CentOS 5.x, 6.x

- Installation Dir: /opt/aspera/orchestrator/
- Config Dir: /opt/aspera/var/config/orchestrator/
- Runtime Dir: /opt/aspera/var/run/orchestrator/
- Archive Dir: /opt/aspera/var/archive/orchestrator/
- Log Dir: /opt/aspera/var/run/orchestrator/log/orchestrator.log
- License file locations: /opt/aspera/orchestrator/config/<client_id>-orchestrator.aspera-license or /opt/aspera/var/config/orchestrator/licenses/<client_id>-orchestrator.aspera-license

Operating Systems: Windows 2003 Server, Windows 2008 R2 Standard and Windows 2012 Server

- Installation Dir: C:\Program Files (x86)\Aspera\Orchestrator\www
- Config Dir: C:\Program Files (x86)\Aspera\ Orchestrator\www\var\config\
- Runtime Dir: C:\Program Files (x86)\Aspera\Orchestrator\www\var\run\
- Archive Dir: C:\Program Files (x86)\Aspera\Orchestrator\www\var\archive\
- Log Dir: C:\Program Files (x86)\Aspera\Orchestrator\www\var\run\log\orchestrator.log
- License file locations: C:\Program Files (x86)\Aspera\Orchestrator\www\config\<client_id>-orchestrator.aspera-license or
  C:\Program Files (x86)\Aspera\Orchestrator\www\var\config\licenses\<client_id>-orchestrator.aspera-license

OPERATIONS

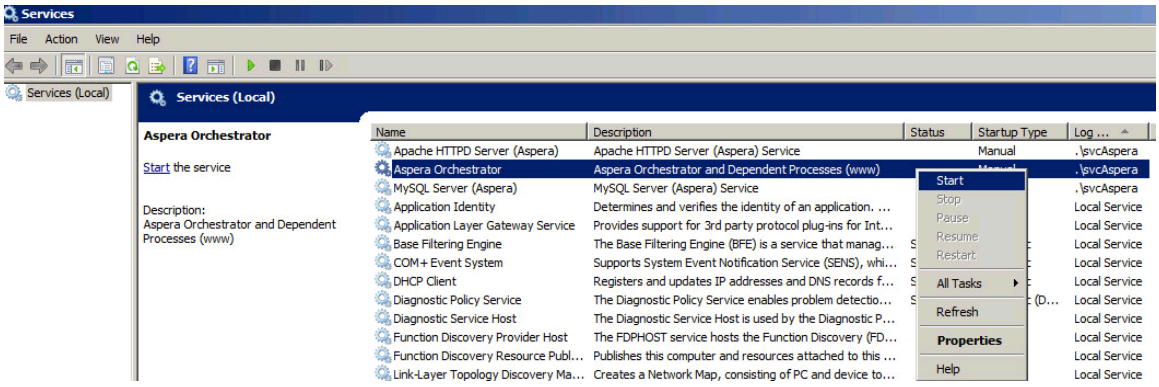
HOW TO START ORCHESTRATOR

Note: Orchestrator does not mount mapped drives being used in the workflows (if any). So before you start Orchestrator please make sure all the required drives are mounted/mapped with a valid username and password.

STEPS TO START FROM SERVICES MENU IN WINDOWS

1) Click the Start button for Windows
2) Navigate to Administrative Tools -> Services menu item or Search for “services.msc” or Locate the Run command and type “services.msc” and press Enter command

3) Locate the service Aspera Orchestrator and right click it
4) Select “Start” to start Orchestrator
    a. Note that Aspera Orchestrator needs MySQL Server to be started



STEPS TO START ORCHESTRATOR FROM WINDOWS COMMAND LINE

If Orchestrator needs to be started manually from the command line

1) Click the Start button for Windows
2) Locate the command line shortcut
3) Right click command line shortcut and start it as Administrator
4) Change directory to Orchestrator directory using the following command

```
# cd "C:\Program Files (x86)\Aspera\Orchestrator\www"
```

5) Check status of Orchestrator using the following command (if any processes are running, go to Step 6, else Step 7)

```
# bin\orchestrator.bat status
```

6) Kill currently running processes using the following command (takes a couple of seconds to kill all running instances)

```
# bin\orchestrator.bat all kill
```

7) Start Orchestrator using the following command

```
# bin\orchestrator.bat
```

STEPS TO START ORCHESTRATOR IN UNIX

1) Login into the UNIX machine that hosts Orchestrator with a user with privileges to start Orchestrator.
2) Start Orchestrator using the following command(s)

```
# asctl orchestrator:start
```

(Or)

```
# /etc/init.d/AsperaOrchestrator start
```

(Or)

```
# service AsperaOrchestrator start
```

HOW TO STOP ORCHESTRATOR

STEPS TO STOP ORCHESTRATOR IN UNIX

1) Login into the UNIX machine that hosts Orchestrator with a user with privileges to stop Orchestrator.
2) Stop Orchestrator using the following command(s)

```
# asctl orchestrator:stop
```

(Or)

```
# /etc/init.d/AsperaOrchestrator stop
```

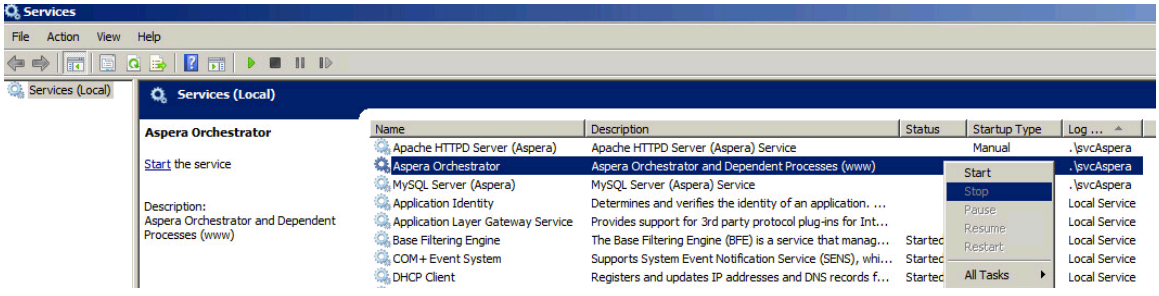
(Or)

```
# service AsperaOrchestrator stop
```

STEPS TO STOP ORCHESTRATOR IN WINDOWS

Method 1:

1) Click on Start button for Windows
2) Navigate to Administrative Tools -> Services menu item or Search for “services.msc” or Locate the Run command and type “services.msc” and press Enter command
3) Locate the service Aspera Orchestrator and right click it
4) Select “Stop” to stop Orchestrator



Method 2:

If Orchestrator needs to be stopped manually from command line

1) Click on Start button for Windows
2) Locate the command line shortcut
3) Right click command line shortcut and start it as Administrator
4) Change directory to Orchestrator directory using the following command

```
# cd "C:\Program Files (x86)\Aspera\Orchestrator\www"
```

5) Check status of Orchestrator using the following command (if any processes are running go to Step 6)

```
# bin\orchestrator.bat status
```

6) Kill currently running processes using the following command (takes a couple of seconds to kill all running instances)

```
# bin\orchestrator.bat all kill
```

HOW TO RESTART ORCHESTRATOR

STEPS TO RESTART ORCHESTRATOR IN UNIX

1) Login into the UNIX machine that hosts Orchestrator with a user with privileges to restart Orchestrator.
2) Restart Orchestrator using the following command

```
# asctl orchestrator:restart
```

(Or)

```
# /etc/init.d/AsperaOrchestrator restart
```

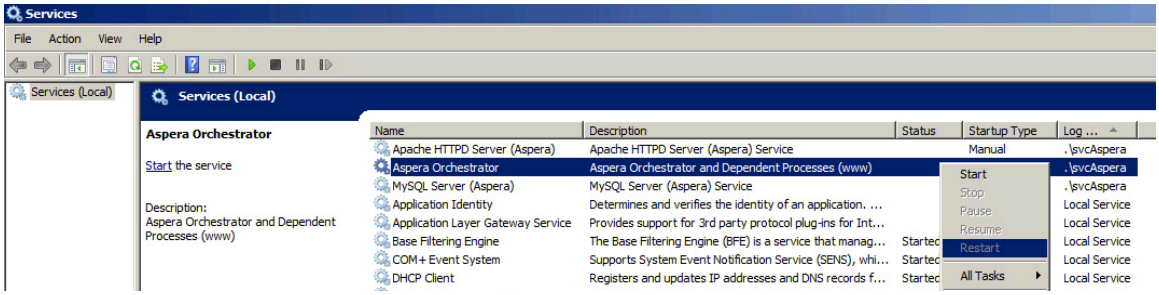
(Or)

```
# service AsperaOrchestrator restart
```

STEPS TO RESTART ORCHESTRATOR IN WINDOWS

Method 1:

1) Click on Start button for Windows
2) Navigate to Administrative Tools -> Services menu item or Search for “services.msc” or Locate the Run command and type “services.msc” and press Enter command
3) Locate the service Aspera Orchestrator and right click it
4) Select “Restart” to restart Orchestrator



Method 2:

If Orchestrator needs to be restarted manually from command line

1) Click on Start button for Windows
2) Locate the command line shortcut
3) Right click command line shortcut and start it as Administrator
4) Change directory to Orchestrator directory by command

```
# cd "C:\Program Files (x86)\Aspera\Orchestrator\www"
```

5) Check status of Orchestrator (if any processes are running go to Step 6)

```
# bin\orchestrator.bat status
```

6) Kill currently running processes (takes a couple of seconds to kill all running instances)

```
# bin\orchestrator.bat all kill
```

7) Start Orchestrator

```
# bin\orchestrator.bat
```

TROUBLESHOOTING

The default location of Orchestrator log can be found under C:\Program Files (x86)\Aspera\Orchestrator\www\var\run\log\orchestrator.log on Windows or /opt/aspera/var/run/orchestrator/log/orchestrator.log on UNIX.

The log file is rotated each day to prevent it from growing too large.

Orchestrator relies on MySQL database functioning properly. If MySQL database is not started, Orchestrator won't start.

1) MySQL database won't start if:
   - The user doesn't have permissions to run MySQL
   - The disk drive is full
   - If MySQL (or the user with which MySQL was started)
     - Doesn't have rights to write into mysql logs
     - Cannot write to the disk with temp files (PID files, SOCK files)

2) If Ruby executable or Ruby Installation under C:\Ruby have been deleted or the user doesn't have permissions to run Ruby
- Ruby and other Ruby related executable(s) have been added to the PATH environment variable. They have to remain added to the PATH variable on this machine.

3) Orchestrator also won't start if the license file is absent or if expired. Look for messages in the log file that indicate this. The footer in the Orchestrator UI also gives an indication of when a license file expires.

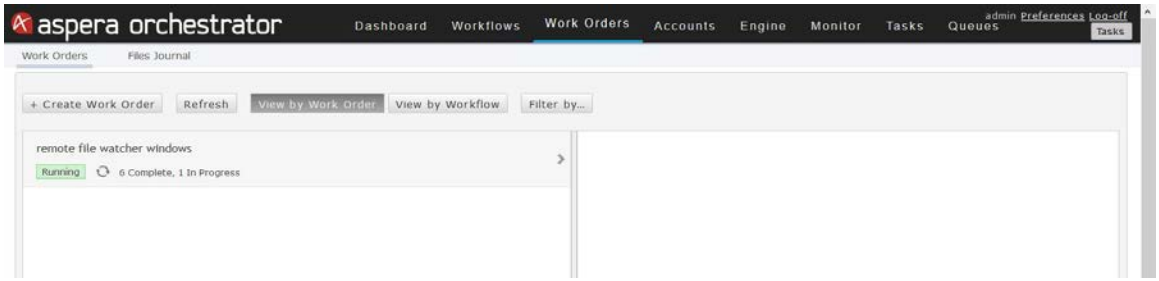
HOW TO CONNECT TO ORCHESTRATOR UI

Enter the following in a web browser (where <ip address> is the Orchestrator node IP address):

http://<ip address>/aspera/orchestrator

In the above window, for initial login use admin/admin

The window that appears after login is the work order page.



A note to Administrators

Please reset the "admin" user password to a secure one or disable the user after creating a different admin user.

Some Best Practice for Admin users:

- Prefix user name with admin_
- An example for Nick's admin user would be: admin_nick and his password

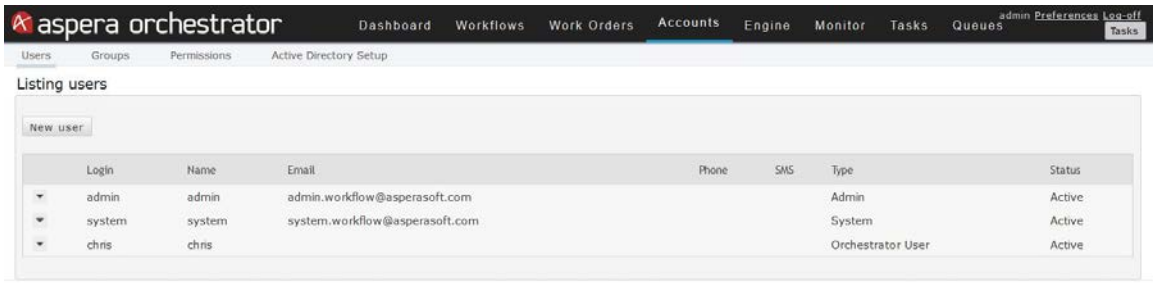
Note: With Orchestrator v2.4.1, strong password policies have been introduced. This is a switch-based feature and can be made active if the '*impose_strong_password*' parameter is set to '*true*' in the orchestrator.yml file. By default this value is false.

If '*impose_strong_password*' is set to '*true'*, after three unsuccessful attempts to log in because of incorrect password, the user account will become inactive and Administrator will have to make the user as active from the GUI manually. The number of unsuccessful login attempts that can be allowed for user comes from '*allowed_logon_attempts*' parameter in orchestrator.yml. By default the value for this parameter is set to three.

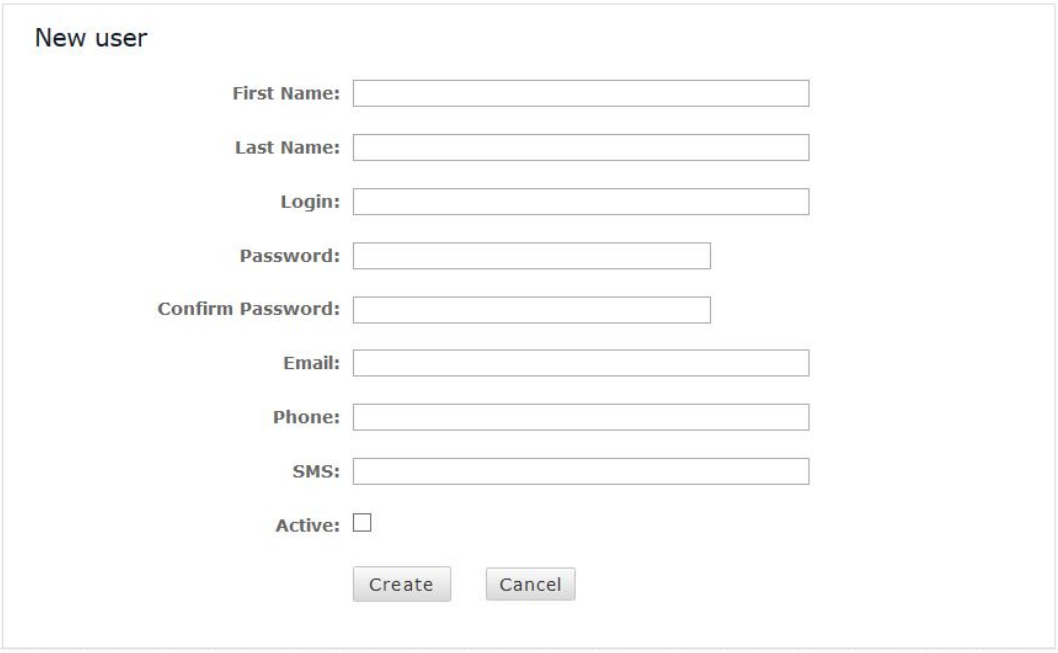
USER ACCOUNTS

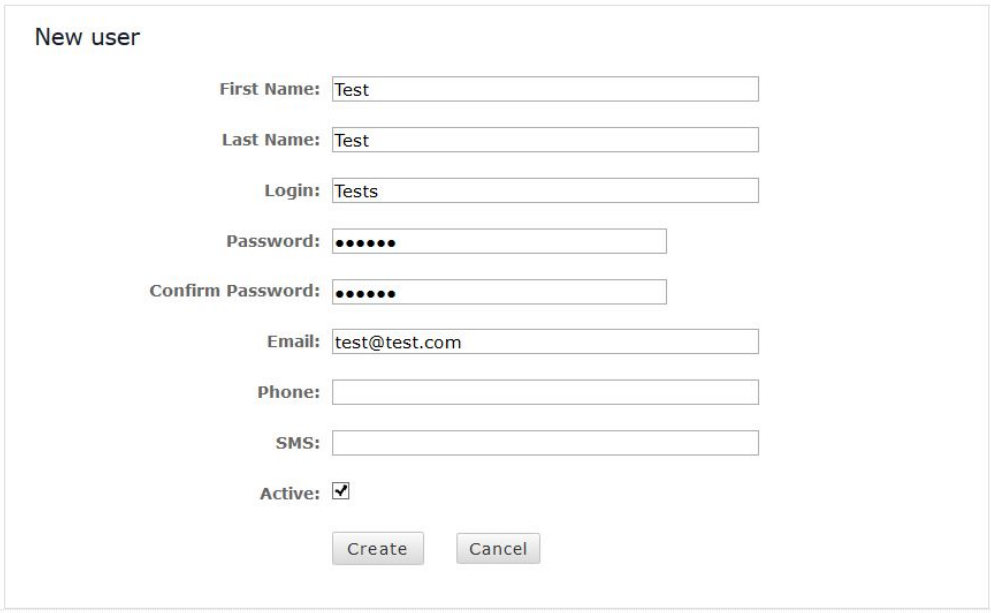
HOW TO CREATE A USER

Navigate to Accounts tab.



The New User button renders a page that allows user creation.





Enter all the information on the above screen. Email, phone and SMS are optional fields. Click on the Active check box once all information is entered and Click Create.

Password policies:

With Orchestrator v2.4.1, strong password policies have been introduced. This is a switch-based feature and can be made active if the '*impose_strong_password*' parameter is set to '*true*' in the orchestrator.yml file. By default this value is false.

With the *impose_strong_password* flag set to true, user needs to enter a strong password that should satisfy the following criteria:

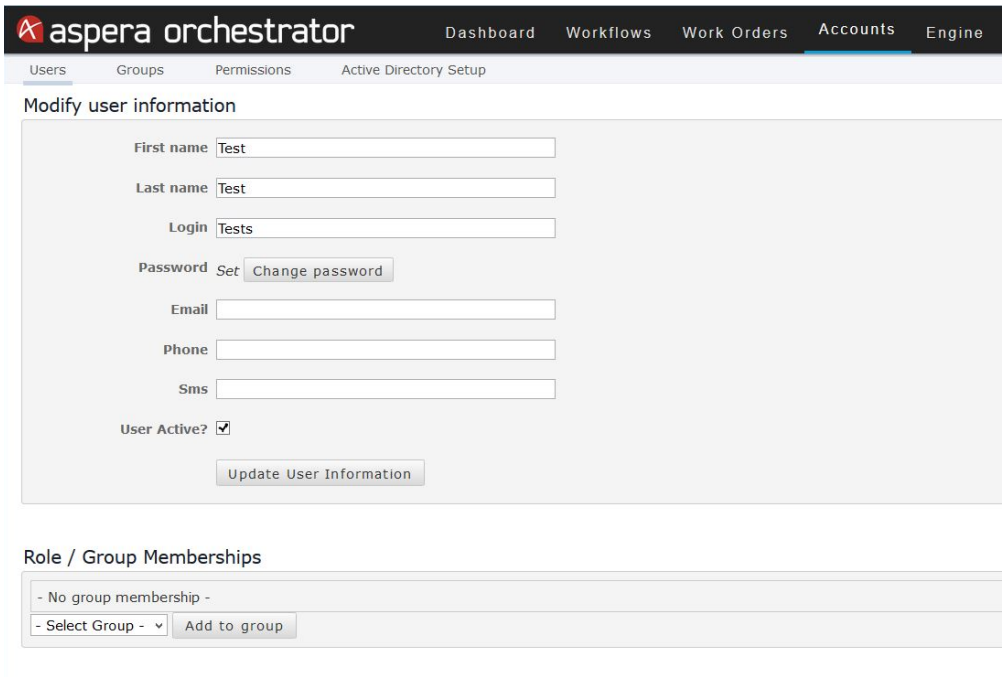
1. Should have minimum 8 character of length.
2. Should have at least one uppercase character
3. Should have at least one lowercase character
4. Should have at least one of these (@#$%^&+=) special characters.

These rules are defined in a regular expression that can be configured or changed if required. The regular expression comes from '*strong_password_regex'* parameter in orchestrator.yml. Modifying parameters of orchestrator.yml is discussed in the later sections of this document.

HOW TO ASSIGN A USER TO A GROUP

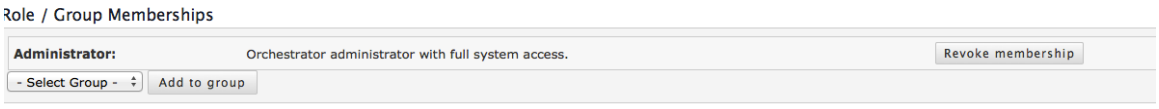
Method 1:

Once the Create button is clicked on a User creation page, Orchestrator allows you to assign the user to a group.



In the drop-down -Select Group-, select a group to which the user needs to be added.

And Click Add to Group

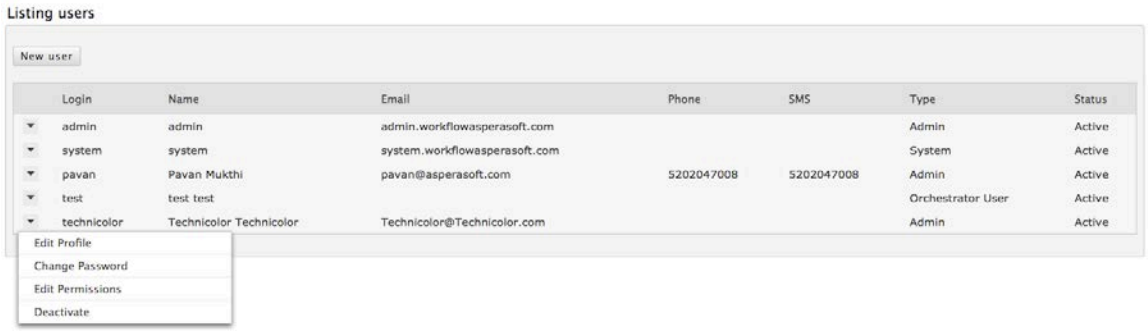


A user must be in the group “Developer” to see the “Workflows” tab in the Orchestrator bar, in the group “System” to see the “engine” tab, in the group “Contributor” to see the “Operator” tab, in the group “Scheduler” to see the “Queues” tab, in the group “Operator” to see the “Work Orders” and “Monitor” tab. Only a user in the group “Admin” can destroy work orders.

Method 2:

Navigate to Accounts tab.

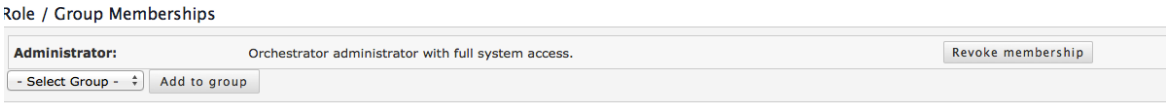
The down arrow beside each user allows more options for each user.



Edit permissions takes you to a page which allows user's groups to be edited.

In the drop-down -Select Group-, select a group to which the user needs to be added.

And Click Add to Group



Find that in the User preferences the new group is added to the list of the user's group.

HOW TO CHECK THE USER PREFERENCES

Navigate to Accounts tab.

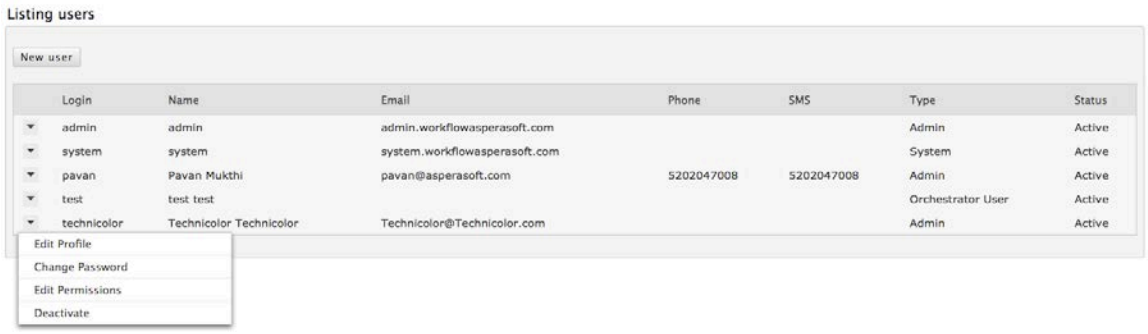
The down arrow beside each user allows more options for each user.

Edit Profile take you to a page to update User profile

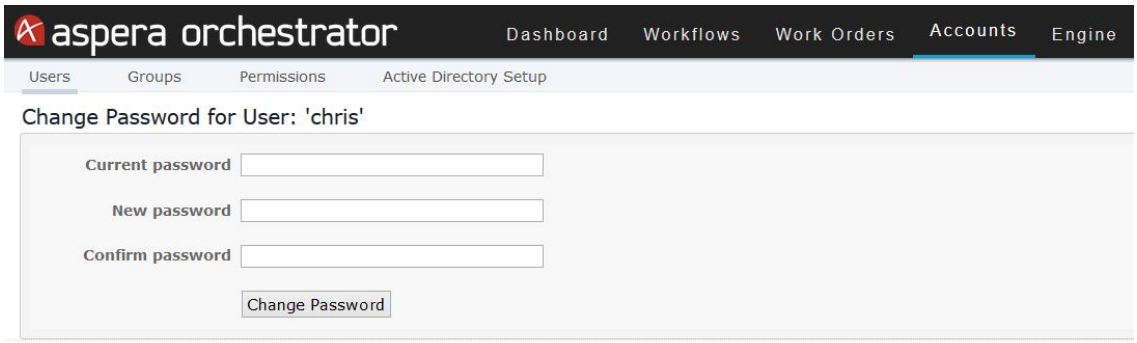
HOW TO RESET THE PASSWORD

Navigate to Accounts tab.

The down arrow beside each user allows more options for each user.



Click change password and provide data in all the fields



Password policies:

As discussed in the user creation section, with Orchestrator v2.4.1, strong password policies have been introduced. This is a switch-based feature and can be made active if the '*impose_strong_password*' parameter is set to '*true*' in the orchestrator.yml file. By default this value is false.

With the *impose_strong_password* flag set to true, user needs to enter a strong password that should satisfy the following criteria:

1. Should have minimum 8 character of length.
2. Should have at least one uppercase character
3. Should have at least one lowercase character
4. Should have at least one of these (@#$%^&+=) special characters.

These rules are defined in a regular expression that can be configured or changed if required. The regular expression comes from '*strong_password_regex'* parameter in orchestrator.yml. Modifying parameters of orchestrator.yml is discussed in the later sections of this document.

As part of the v2.4.1, if the '*impose_strong_password*' is set to true, users cannot set the new password as any of the previous three passwords.

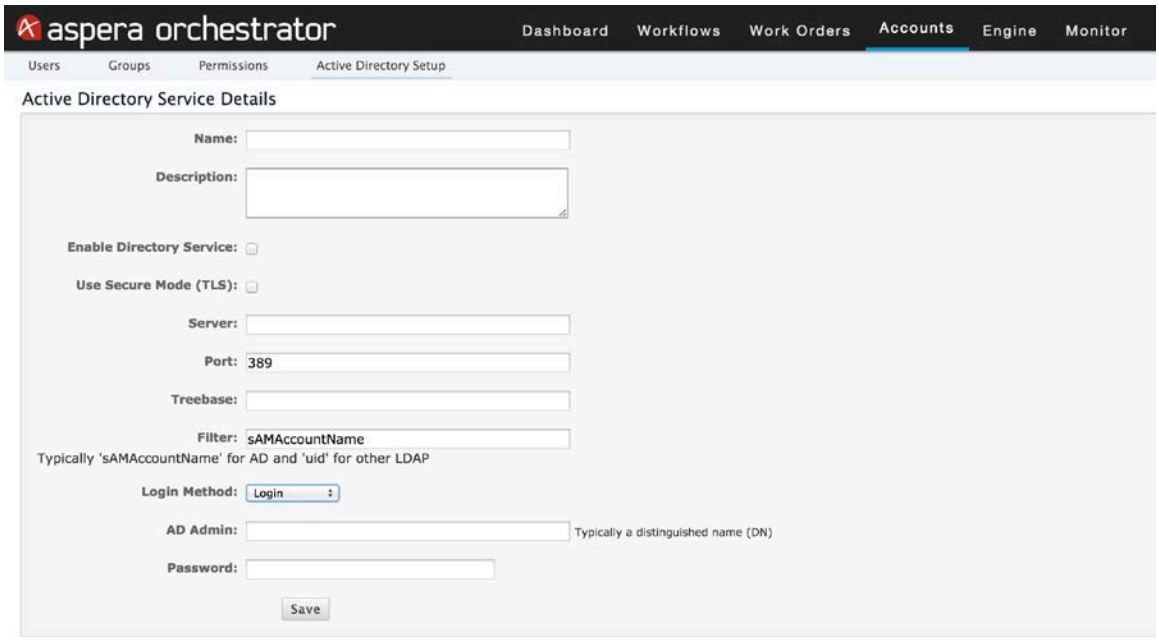
HOW TO CONFIGURE THE LDAP ACCESS

With Orchestrator 2.3.5 and newer releases Orchestrator supports configuration for Multiple Active Directory / LDAP servers to be configured.

Navigate to Accounts tab.

Click on Active Directory Setup tab. Click on the 'New Active Directory' button.

That will navigate to the below screen.



Enter the Name and Description for the new Active Directory.

Click on "Enable Directory Service" to enable the LDAP service.

Enter the LDAP server fully qualified name or IP address in "Server". Enter the LDAP service port in "Port".

Enter the base Distinguished Name in "Treebase". Enter the filter as "uid" for LDAP or "sAMAccountName" for Windows Active Directory service (LDAP based).

Select a login method as anonymous (no login/password required) or login (login/password required).

Click on “Save” to validate the settings against the LDAP server (LDAP bind done with Server, Port and Login credentials). If successful, this message appears: “Saved successfully. Connection OK”.

Next, login to Orchestrator with an LDAP username and password (without creating the user in Orchestrator GUI). A user with type “Active Directory User” is created automatically by Orchestrator the first time this login is successful (check in Users tab).

Upon login with loginname/password, Orchestrator performs a combination of LDAP binding and search. First, it connects and binds to the LDAP server via the login credentials as configured in the window above. Then it searches the LDAP server for an entry corresponding to the filter specified above (e.g. uid=loginname). If the entry exists, it re-binds as that user with the supplied password.

Following operations can be done on all the Active Directory Users that are added on Orchestrator

1. Change password.
2. Update user information
3. Delete User

All the above operations will be performed if ‘*perform_ad_operation*’ flag in the orchestrator.yml is set to *‘true’*.

Following points should be noted for each operation that is performed.

Create User on AD:
- This operation will be performed only if the AD server is SSL enabled (port 636)
- Provide valid CN (‘CN=‘ followed by first name, OU (‘OU=‘ followed ou value) and dn of the Active Directory)

Change password:
- This operation will be performed only if the AD server is SSL enabled (port 636)
- No action on Orchestrator database will be performed, as Orchestrator does not store AD password on Orchestrator DB.

Update user information:
- If the ‘*perform_ad_operation’* flag is false no action on Orchestrator database will be performed, as updating user information on Orchestrator DB will make them inconsistent with that on AD.

Delete User:

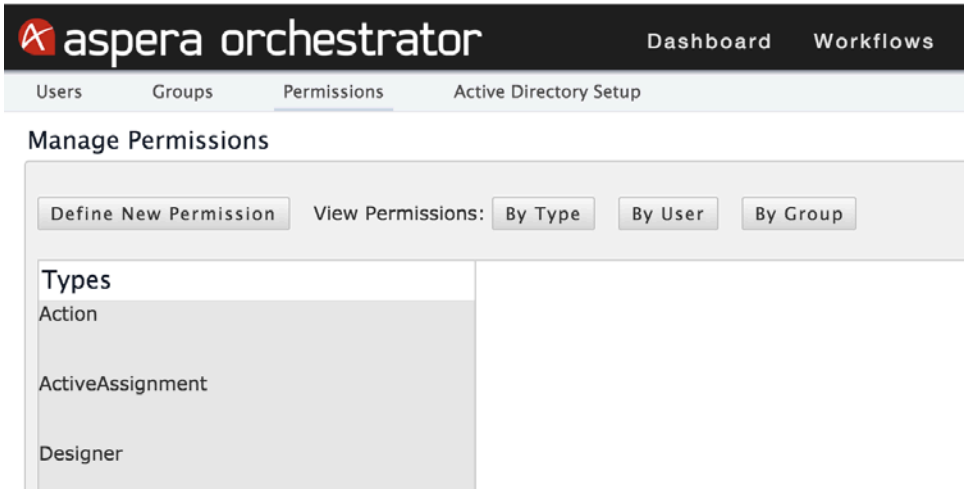
If the ‘*perform_ad_operation’* flag is false, user will be deleted from Orchestrator DB. (User is still active on AD, so can be created again on Orchestrator by logging in with valid credentials).

PERMISSIONS

The Permissions page displays list of permissions for various parts of Orchestrator. The permissions can be listed:

1. By Users
2. By Groups
3. By Types.

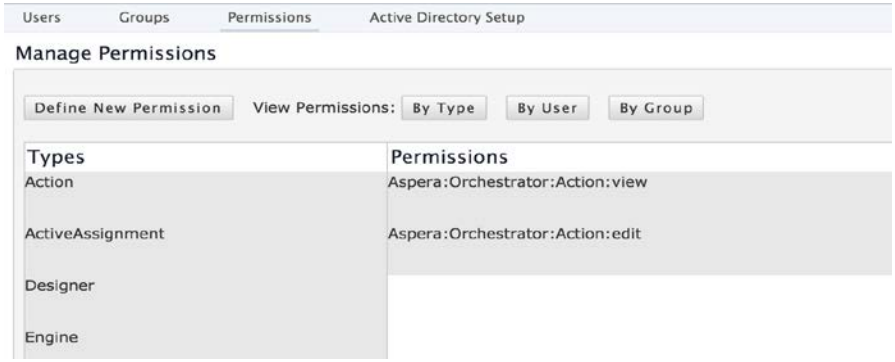
A new permission can also be defined from this page.



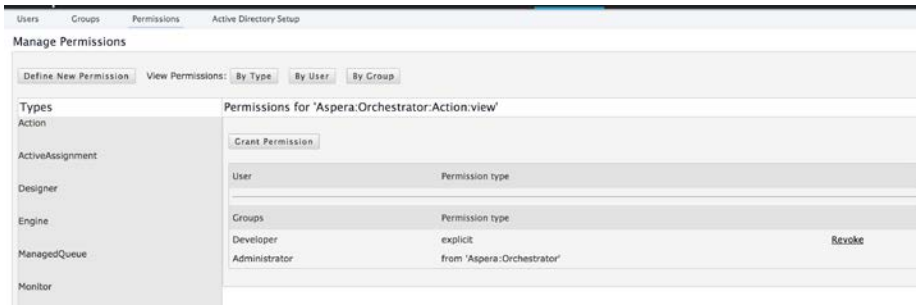
HOW TO SEARCH AND GRANT PERMISSIONS

By default, the permissions are listed by Type.

1. The Left pane displays a list of all Types in the Orchestrator
2. Select the type to view all permissions under it.



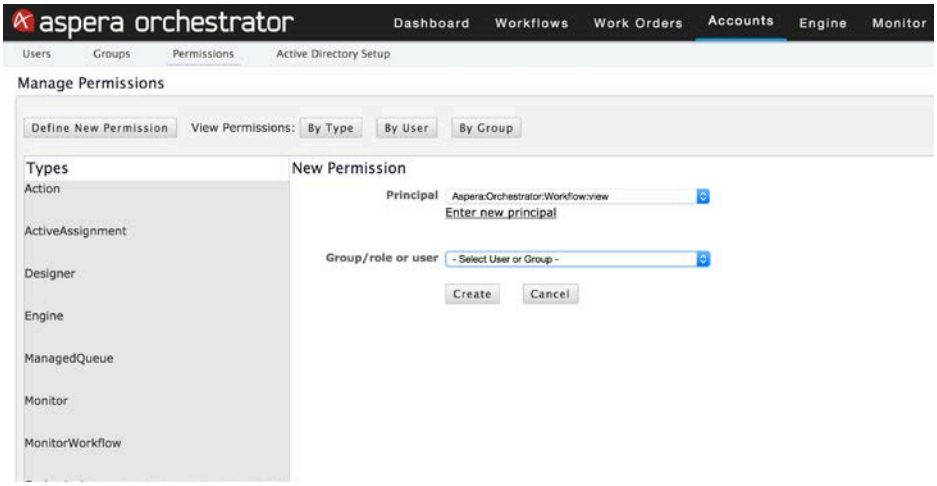
3. Select the permission to view permission details and also to grant that permission to a user or a group.



Similarly, permissions can be searched and granted by user and by group.

HOW TO DEFINE NEW PERMISSION

1. Click the "Define New Permissions" button above the Left pane.



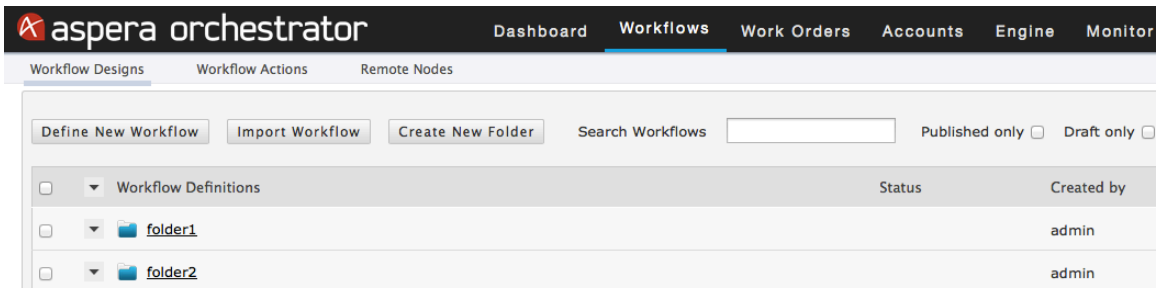
2. Select the principal for the new permission or enter a new principal. Principal is a unique string to identify permissions. E.g.:
   - The Principal 'Aspera:Orchestrator' can be used to set permissions for the entire application.
   - The Principal 'Aspera:Orchestrator:Workflow' for all Workflows in the application
   - The Principal 'Aspera:Orchestrator:Workflow:view' for only view permissions for all Workflows.

3. Select the User or group for the new permission.

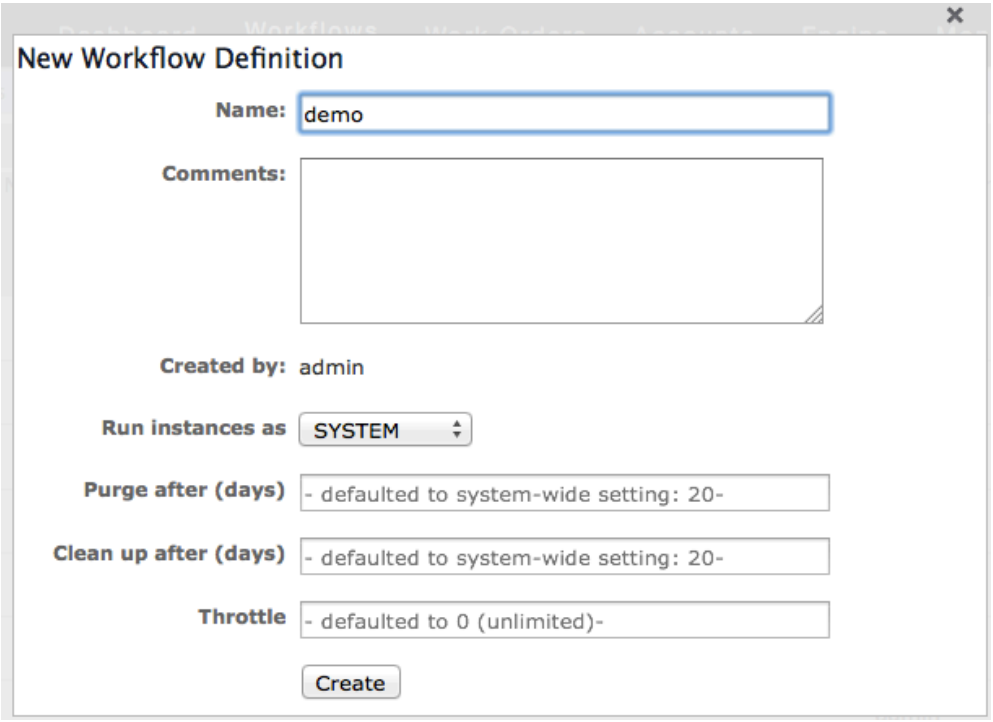
WORKFLOW DESIGNER

HOW TO CREATE A WORKFLOW

Navigate to Workflows tab.



Click on "Define New Workflow" to find the pop up with text fields



Once the name and comments and other fields are entered/chosen clicking on Create will take you to the designer screen.

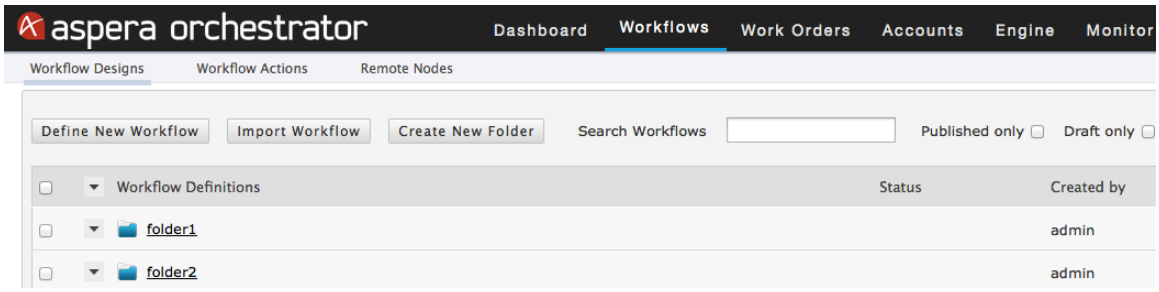
By default, a workflow is created in the root folder (the one you reach when you click on the "Workflows" tab). The workflow can be created in a specific workflow folder rather than in the root folder. Go into the folder where you want to create the workflow, and then click on "Define New Workflow".

The workflow can also be moved manually to a specific folder by selecting the "Move to folder" option after clicking on the down arrow next to the workflow name.

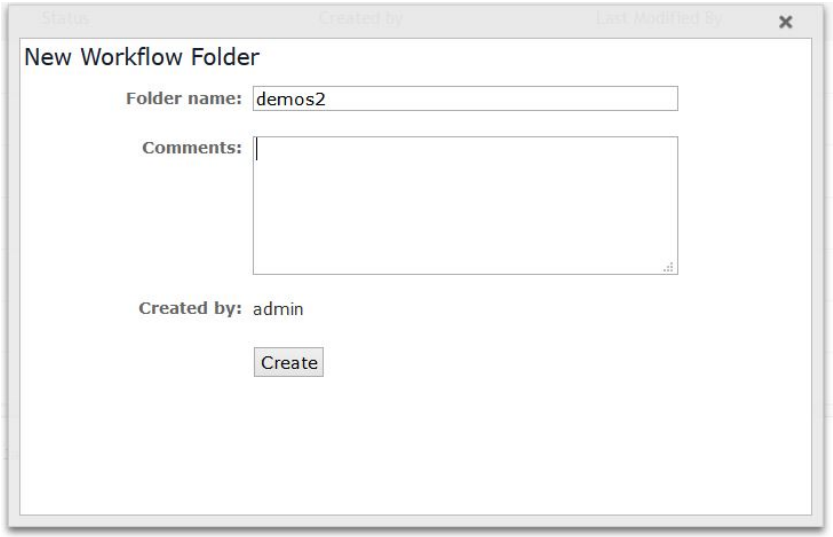
The purge after and clean up after fields allow users to input personalized number of days for the respective activity instead of the system level setting. For example, test workflows and sample workflows can have low value for purge after and clean up after days but production workflows can have higher values for purge after and clean up after days.

HOW TO CREATE A WORKFLOW FOLDER

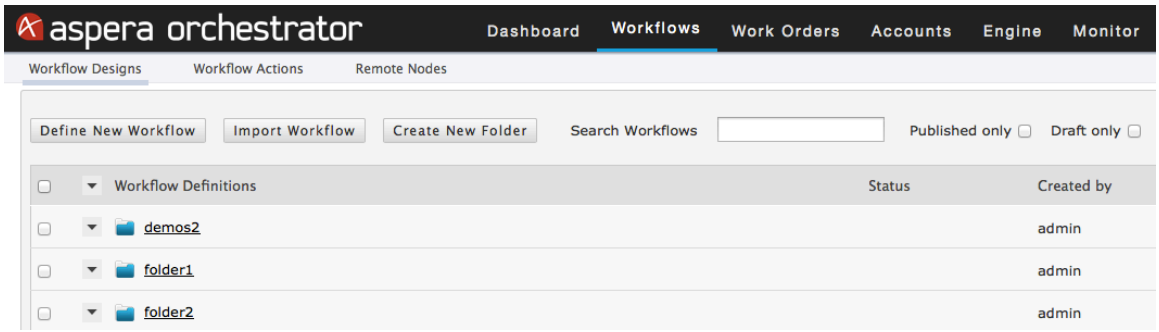
Navigate to Workflows tab. Go into the workflow folder where you want the new folder to be inserted in.



Click on “Create New Folder”.



Type in the folder name and click on “Create”

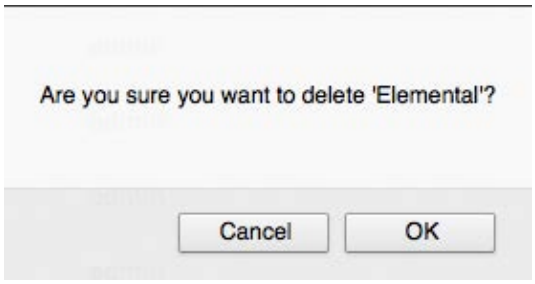


The new folder appears in alphabetical order within the list of folders.

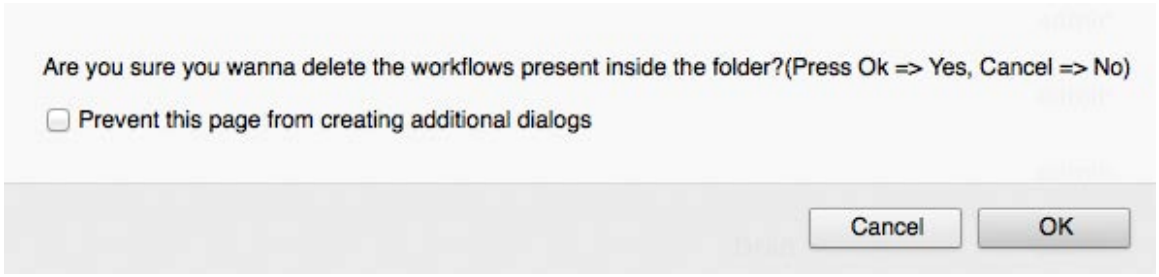
HOW TO DELETE A WORKFLOW FOLDER

Navigate to Workflows tab. Click on the down arrow beside each folder icon. Select “Delete”. Confirm the deletion.

Note: If there are any workflows inside the folder, they will be assigned the parent directory.



Pressing on OK will bring up another popup asking about the contents of the folder.

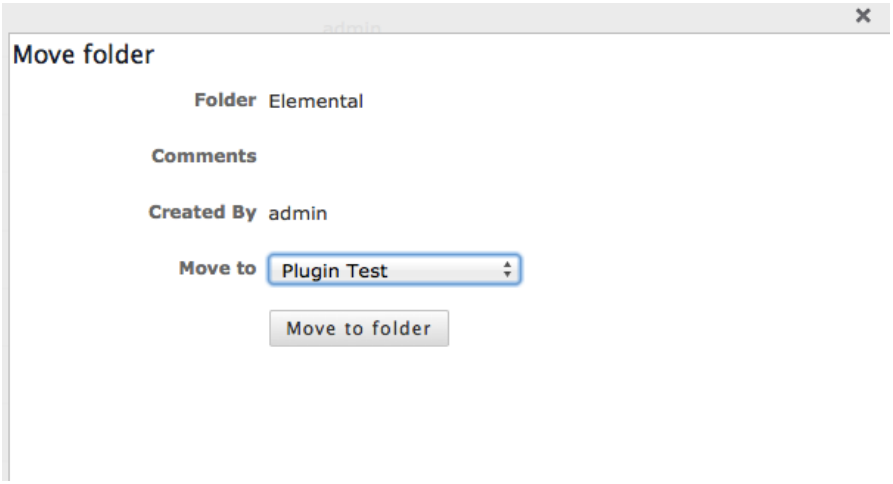


HOW TO MOVE A WORKFLOW FOLDER

Navigate to Workflows tab. Click on the down arrow beside each folder icon. Select “Move to Folder”

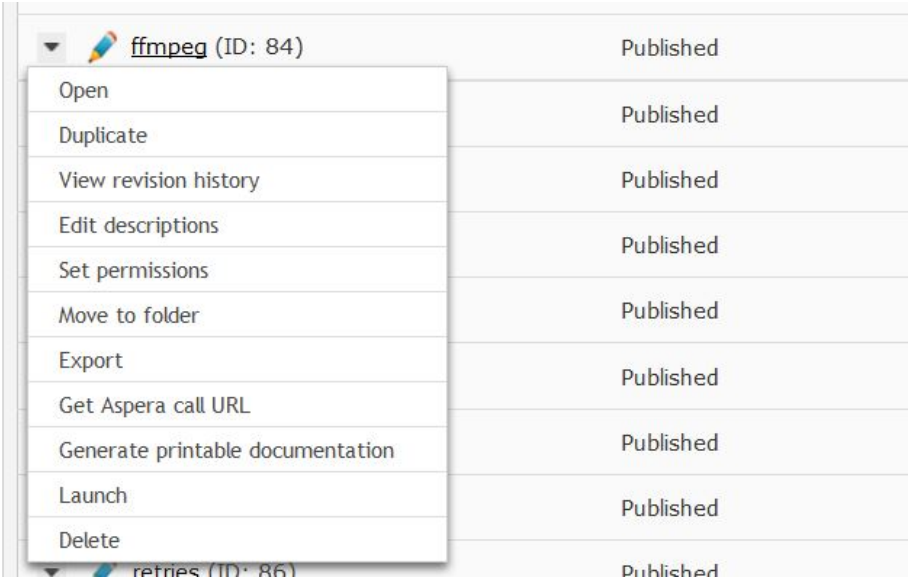
In the popup that appears, select the folder you want to move this folder into from the “Move to” dropdown. The selected folder appears beside the Move to text.

Click Move to Folder to complete the move.

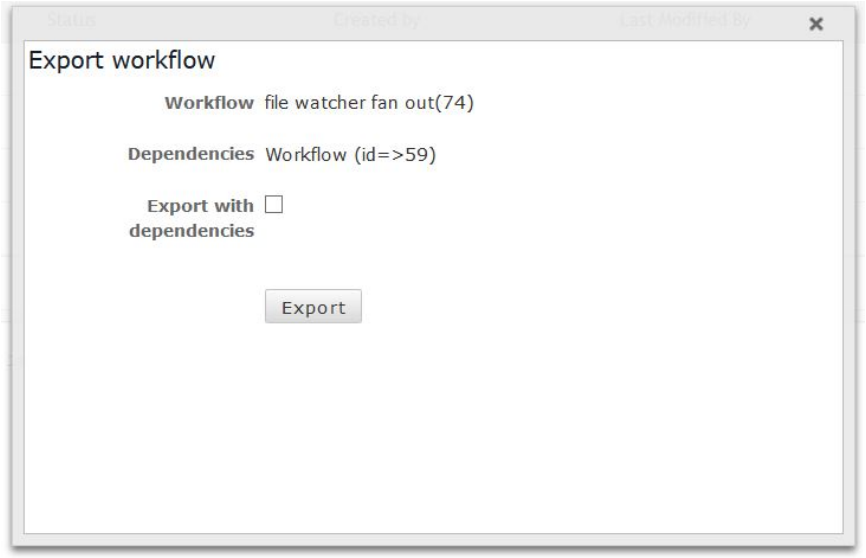


HOW TO EXPORT A WORKFLOW

Navigate to Workflows tab. Go into the folder where is located the workflow to export.



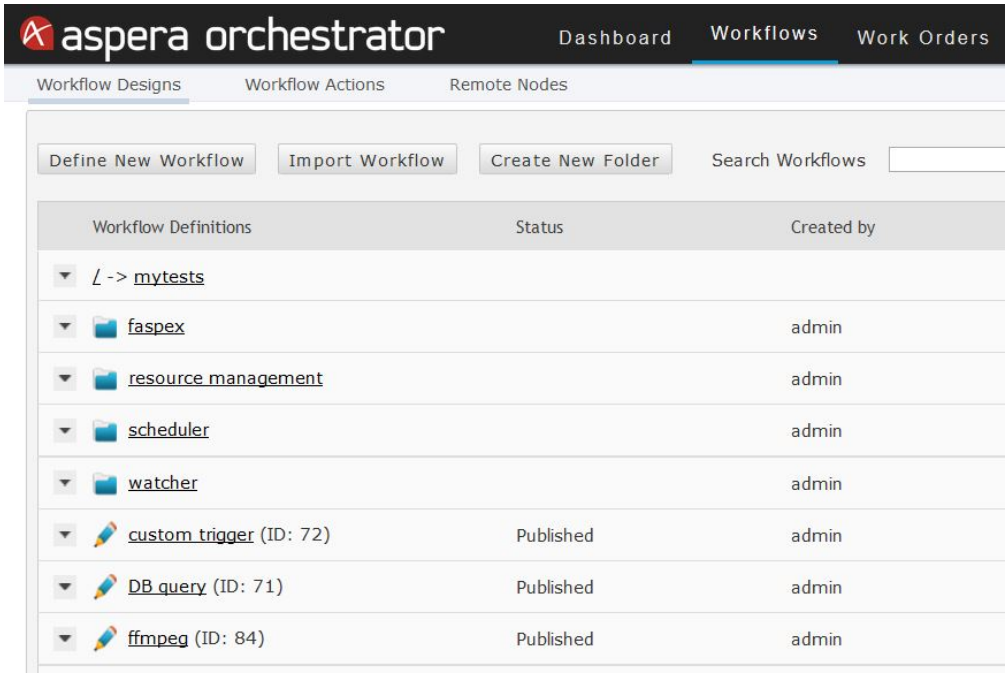
Click on the down arrow next to the workflow name. Select “Export”.



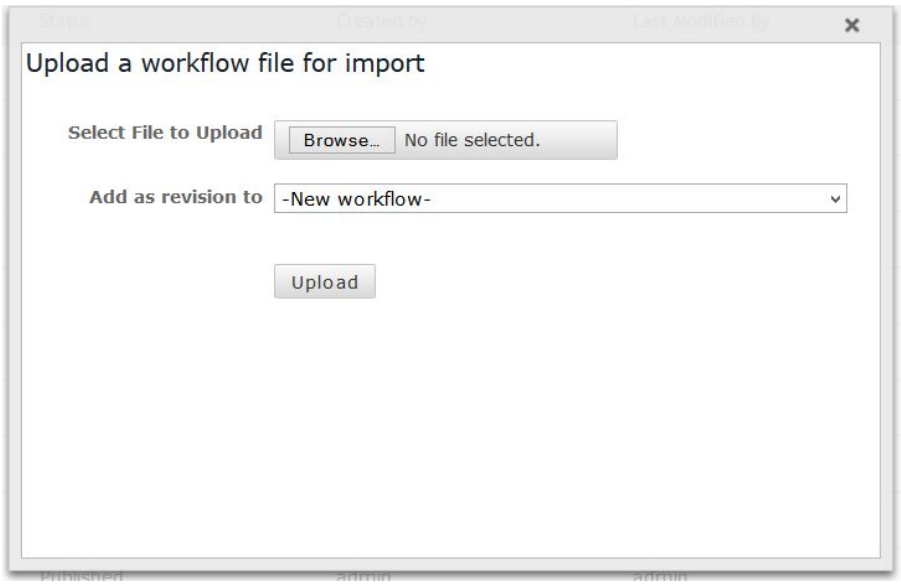
Check the “Export with dependencies” box if the workflow has dependent workflows and you want them exported together. Click on “Export”. The workflow(s) are exported in a single file on your local disk.

HOW TO IMPORT A WORKFLOW

Navigate to Workflows tab. Go into the workflow folder where you want the imported workflow to be inserted in.



Click on “Import Workflow” button

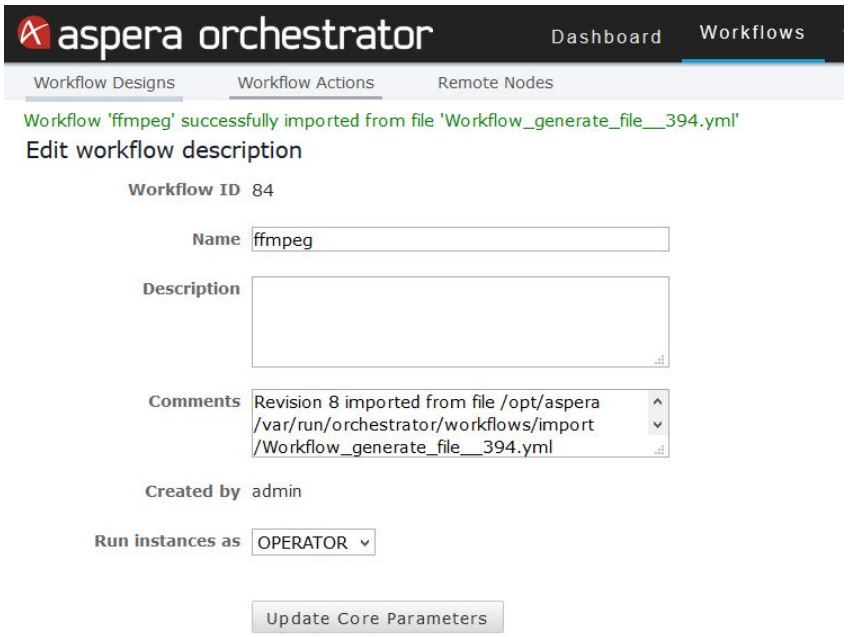


Select the file to be imported from your local disk.

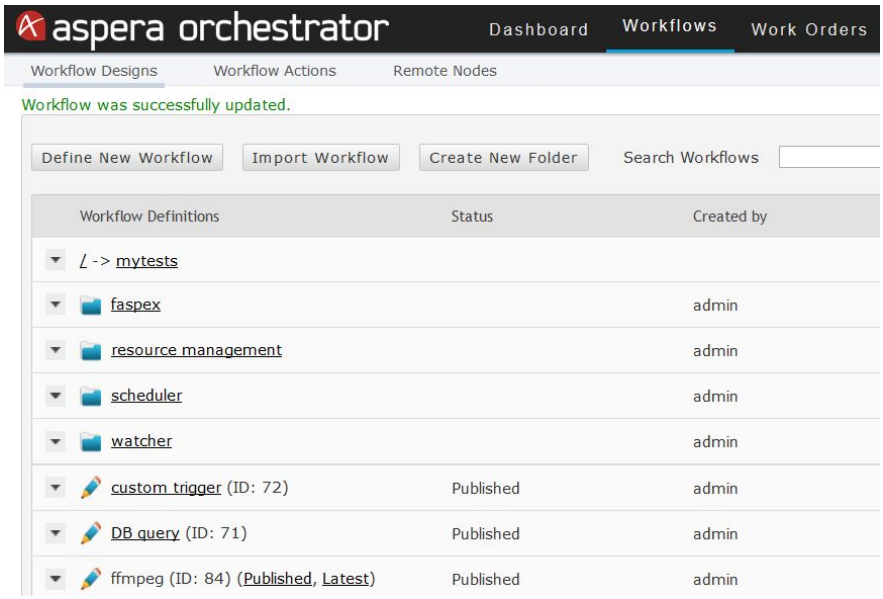
Link to Existing Sub-Workflows? – This check box is added to avoid duplicates while importing the workflows. If you have a sub-workflow that is used by multiple workflows or called multiple times in the same workflow then by checking this option, it checks if the sub-workflow is present already. If yes, it maps the workflow to the existing one, if not it creates 1 single copy and refers the rest to the same.

If you have Journals set-up in the workflow being imported, then the JournalBook would get automatically created and activated if its not already present.

Select a workflow in "Add as revision to" if you want the imported workflow to be added as the latest revision of an existing workflow. Click on "Upload".



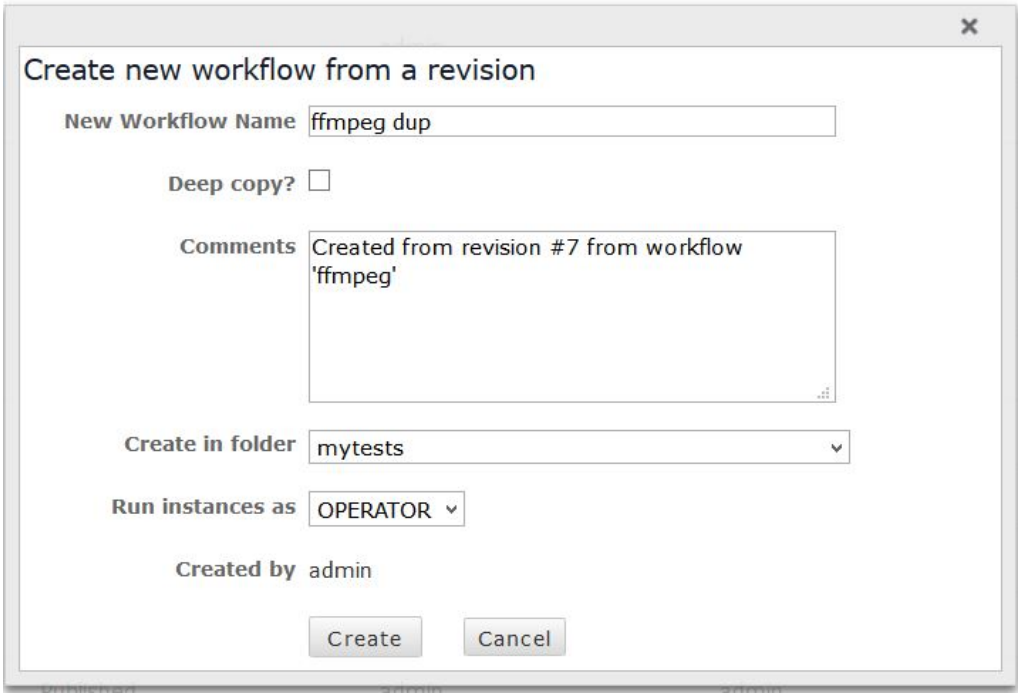
Add a description or comments if necessary. Click on "Update Core Parameters" to take into account the updated information.



Above, the imported workflow has been added as a revision of the ffmpeg workflow. Click on "Latest" next to the workflow name to view the imported workflow.

HOW TO DUPLICATE A WORKFLOW

Click on the down arrow next to the workflow name. Select “Duplicate”.

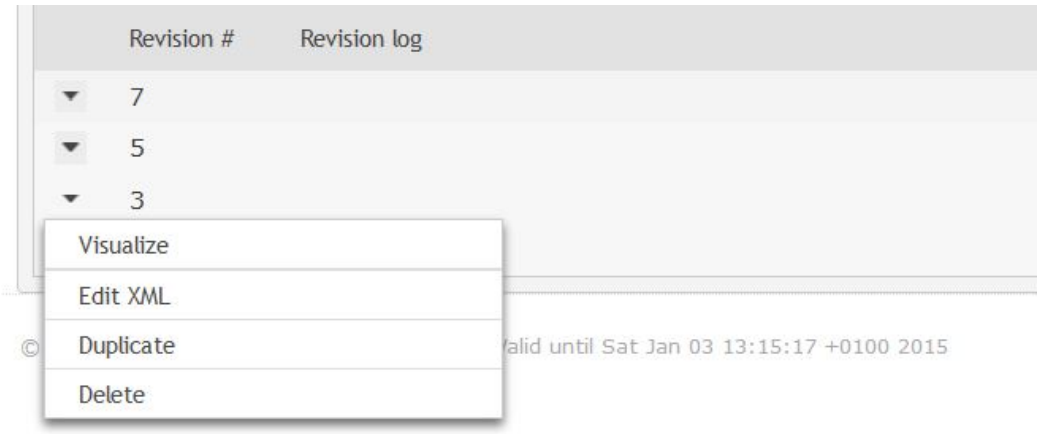


Give a new workflow name and the folder where the workflow will be placed. Click on “Deep copy” to create copies of templates used within the workflow. The templates become configurations in the duplicated workflow.

HOW TO MANAGE WORKFLOW REVISIONS

Click on the down arrow next to the workflow name. Select “View revision history”.

Select the workflow revision you are interested in by clicking in the down arrow next to the revision number.



Select "Visualize" to view and edit the workflow revision. After publishing this revision, it will become the "Published" (executable) version of the workflow.

Select "Edit XML" to view and edit the XML code associated to the workflow revision.

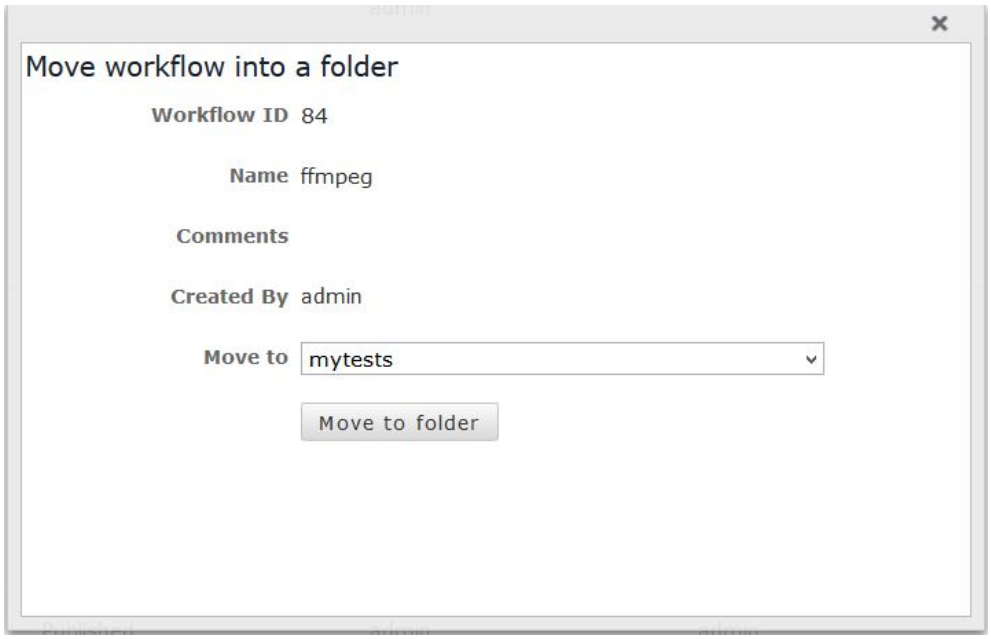
Select "Duplicate" to duplicate the workflow revision.

Select "Delete" to delete the workflow revision.

Note that the templates/configurations are not under revision control. Only the latest template/configuration version is visible in any workflow revision.

HOW TO MOVE A WORKFLOW IN A FOLDER

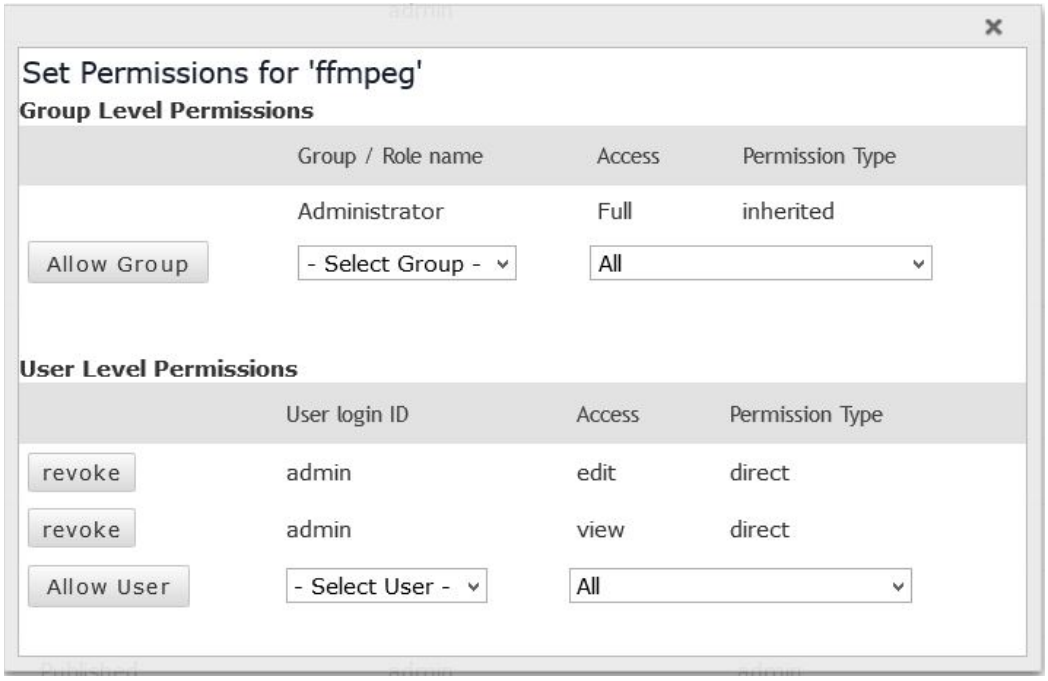
Click on the down arrow next to the workflow name. Select "Move to folder"



Select the folder name where the workflow must be moved to. Click on "Move to folder".

HOW TO SET THE WORKFLOW PERMISSIONS

Click on the down arrow next to the workflow name. Select “Set permissions”.



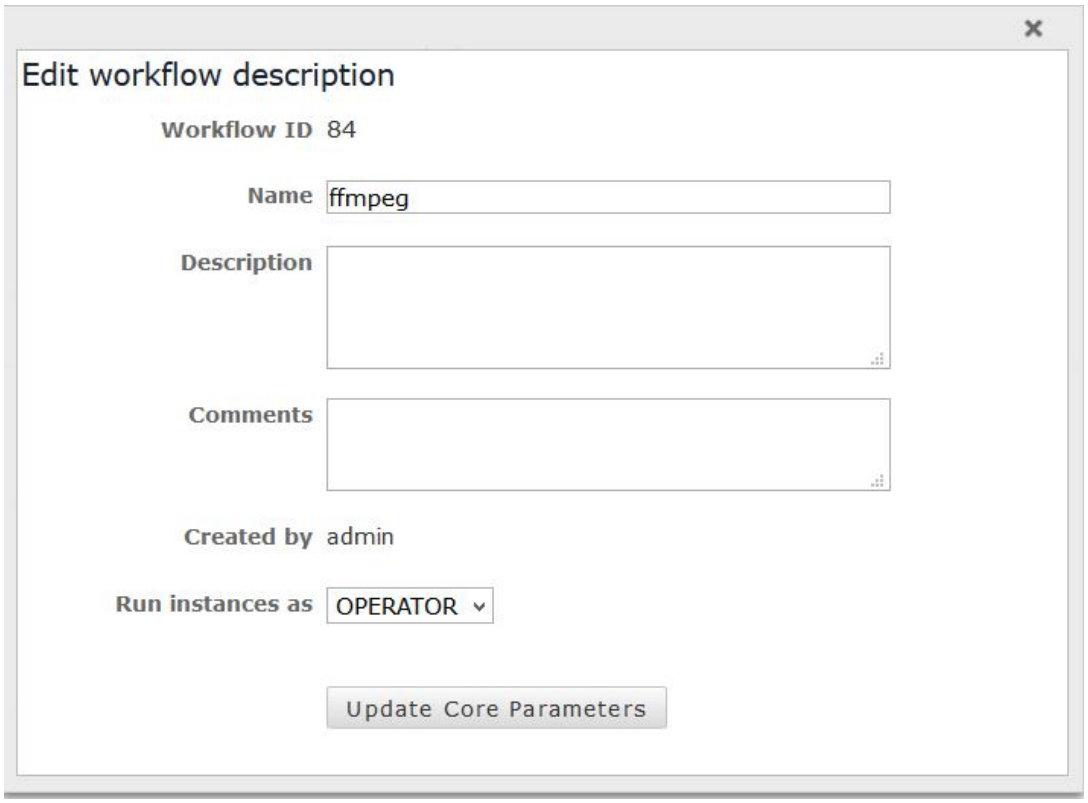
For group level permissions, select the group and permissions (all, view, edit, run). Click on “Allow Group” to add it. Click on “revoke” to remove a group level permission.

For user level permissions, select the user and permissions (all, view, edit, run). Click on “Allow User” to add it. Click on “revoke” to remove a user level permission.

On each workflow you want to monitor in your user dashboard (e.g. workflow stats and step stats), set run permissions at the group or user level.

HOW TO CHANGE THE WORKFLOW DESCRIPTION

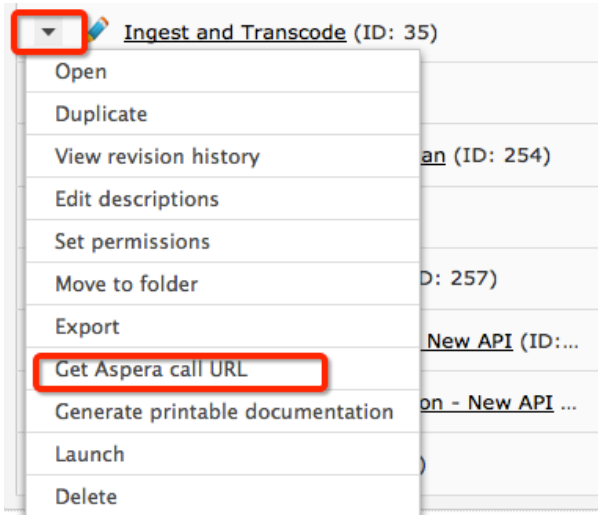
Click on the down arrow next to the workflow name. Select “Edit descriptions”.

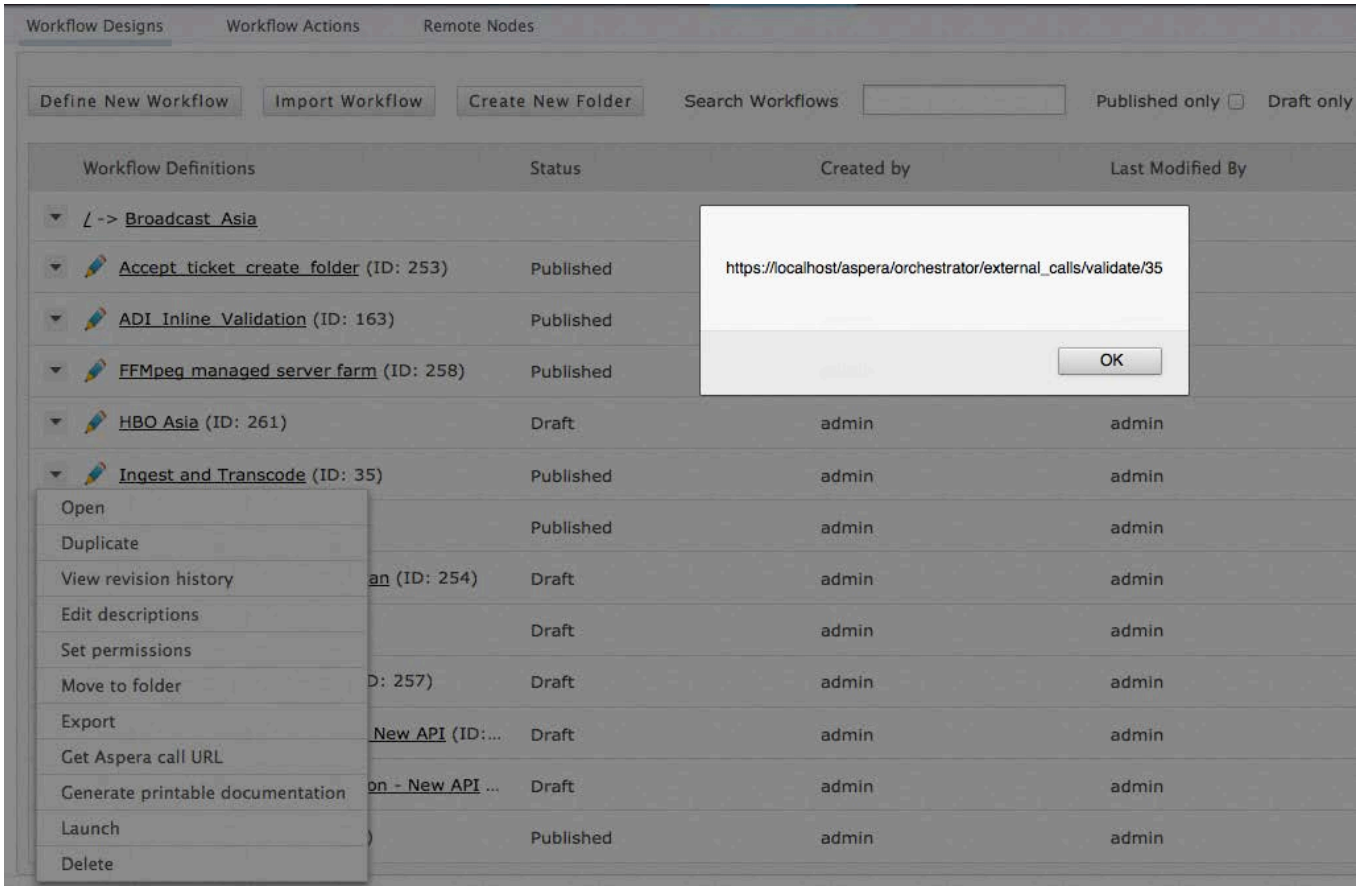


Change the workflow name if necessary. Change the description and/or comments if necessary. Click on “Update Core Parameters” to apply the changes.

HOW TO GENERATE A WORKFLOW CALL URL

Click on the down arrow next to the workflow name. Select "Get Aspera call URL".





Use the generated URL to call the workflow within an ASCP transfer. For example, anti-virus workflow logic could be called inline during a FASP transfer with transfer cancellation if a virus is detected.

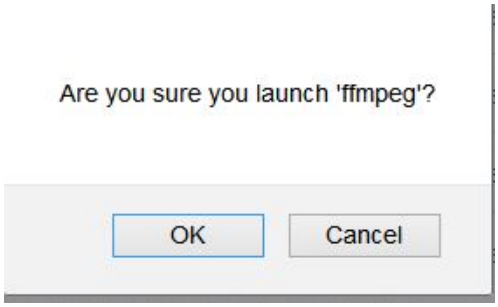
HOW TO PRINT A WORKFLOW

Click on the down arrow next to the workflow name. Select “Generate printable documentation”.

An HTML page is generated with the workflow graphical steps, input/outputs, configuration, comments, and revision history. Print the HTML page with your browser print utility.

HOW TO LAUNCH A WORKFLOW

Click on the down arrow next to the workflow name. Select “Launch”.

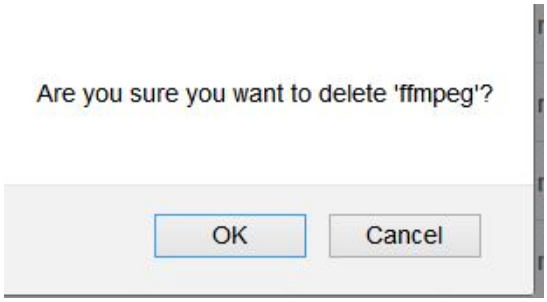


Click on OK to launch the workflow. Navigate to the Work Orders tab to visualize the workflow execution.

The same workflow can also be started from the Work Orders tab. Refer to the HOW TO CREATE A WORK ORDER section under Work Orders.

HOW TO DELETE A WORKFLOW

Click on the down arrow next to the workflow name. Select “Delete”.



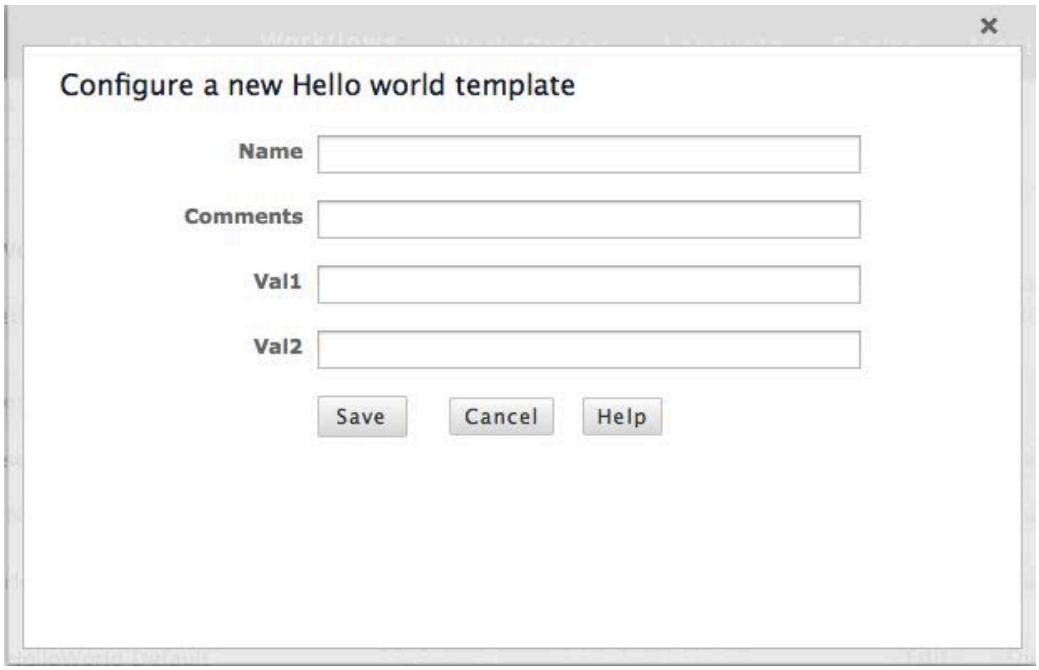
Click on OK to delete the workflow.

WORKFLOW ACTIONS

HOW TO CREATE A TEMPLATE

Select workflow menu, then click the Workflow Actions sub menu to see a list of different plugins/actions on the left hand side of the page. Select any of the Action to see its corresponding action templates. If a plugin is not to be found in the list, navigate to Engine -> Plugins to see if it is enabled.

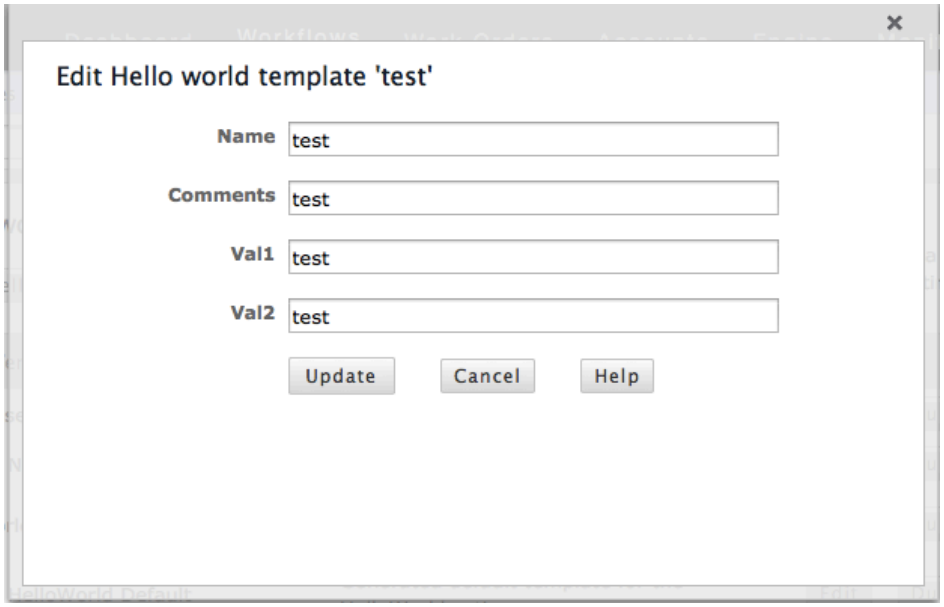
Next, click "New <Action name>" button for the popup asking for information to create a new action template.



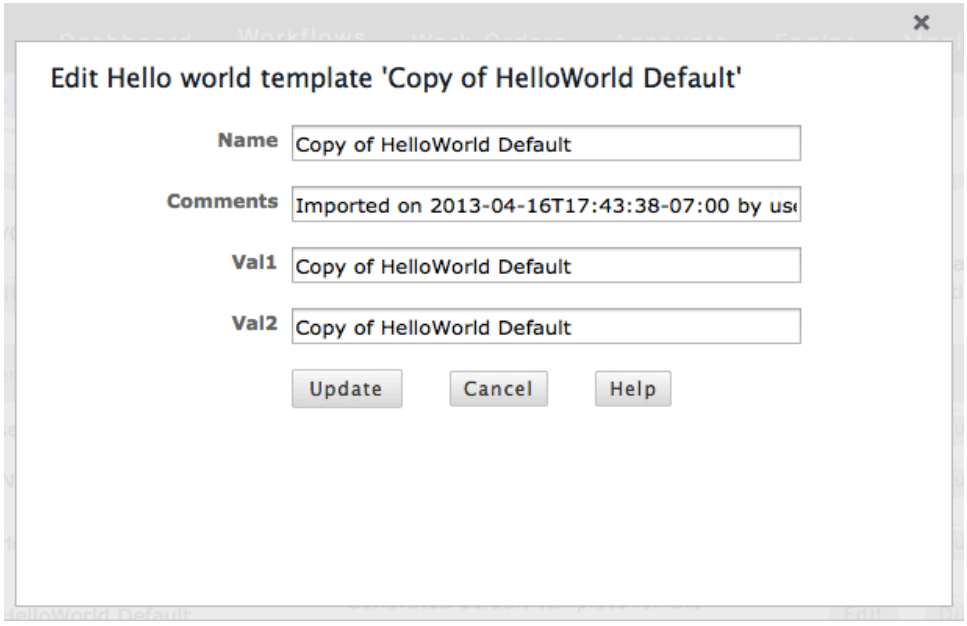
Enter necessary information and click Save. Some of the typical fields in the action template are template name, IP address, source and destination paths,

HOW TO EDIT A TEMPLATE

Click on the "Edit" button that is displayed for every template instance. This will open a dialogue box that will help with editing the template.

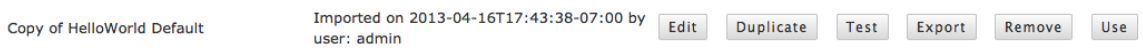


We can also edit a template by clicking on the template; this will open a dialogue box. Then click the "Modify Template" button.



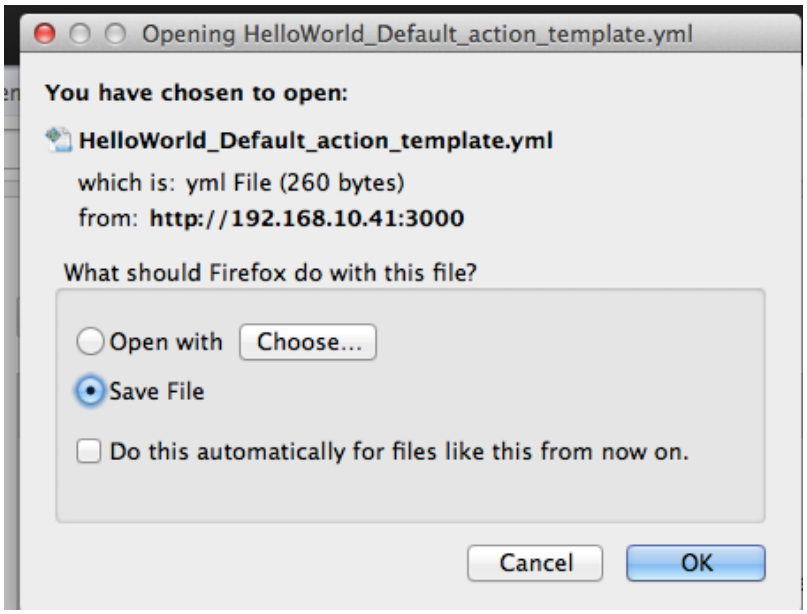
HOW TO DUPLICATE A TEMPLATE

To duplicate a template, click on the "Duplicate" button and a new template with the following name will be created, "Copy of <template name>".



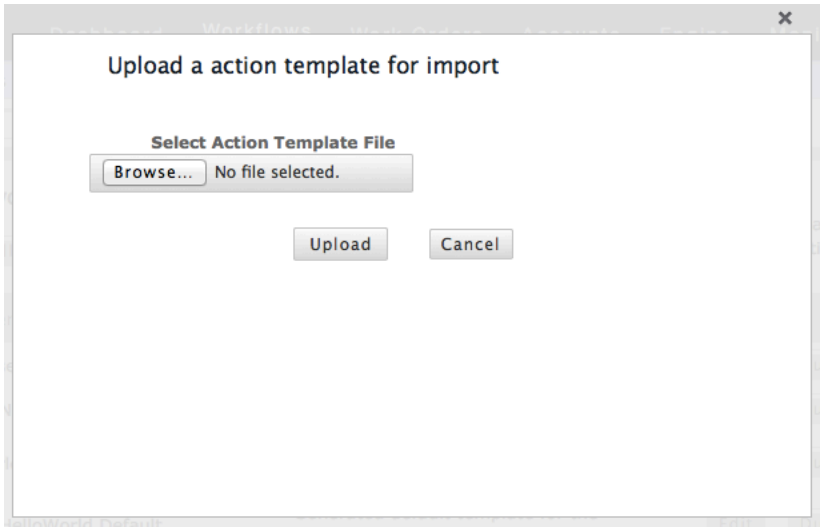
HOW TO EXPORT A TEMPLATE

To export a template, click the “Export” button for a given template record. This will open a browser download dialogue box asking for options to save/open the file.



HOW TO IMPORT A TEMPLATE

Click the “Import Template” button that will open a dialogue box with import options such as Browse from file system.



HOW TO REMOVE A TEMPLATE

Click the “Remove” button seen on the template rows to delete or remove a template instance. This will ask the user for confirmation to remove this instance.

REMOTE NODES

Importance of creating remote nodes:

A remote node consists of information to connect via a FASP Session, ASCMD, SSH or TCP to a remote server. All plugins provide a dropdown to select this remote node for their plugin operations.

Without a remote node users will have to input IP address or Hostnames, user id and password at multiple places in their workflows. If this information changes users are tasked with changing the information in multiple places.

If instead, a user uses a remote node concept and populates the information there, the only place the information changes in that node.

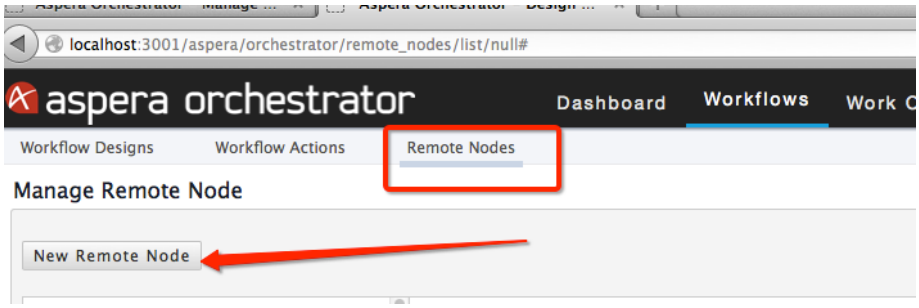
Types of Remote Nodes:

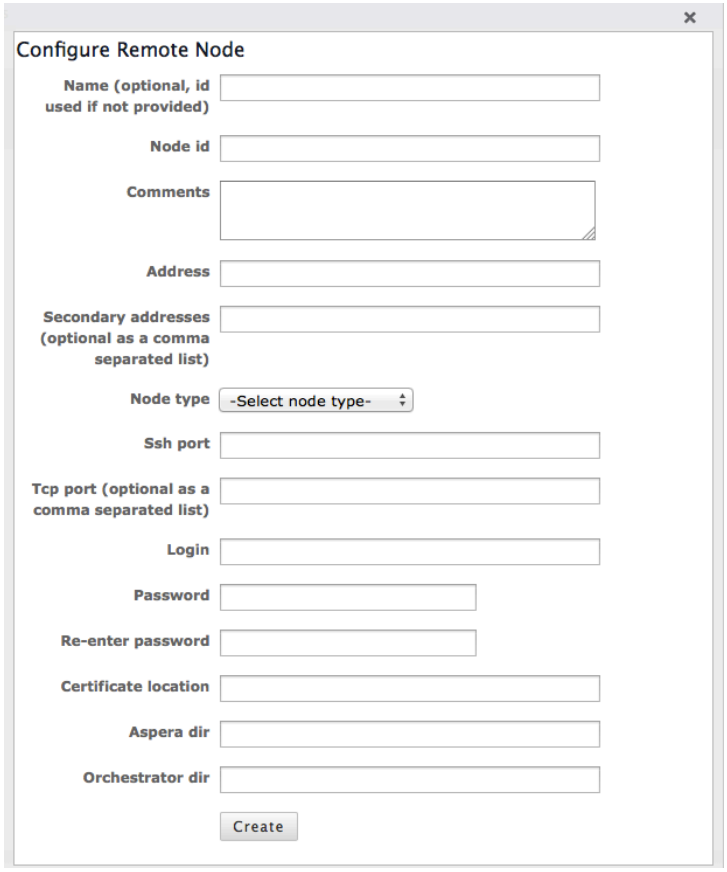
There are 4 types of remote nodes

Node type	Description
Aspera FASP	Use this option when the remote node has one of the following installed a) Aspera P2P b) Aspera Enterprise server c) Aspera Connect server d) Aspera On-Demand server
SSH	Use this option when the remote node accepts SSH connections (*NIX machines accept SSH connections by default, on Windows usually when software like Open SSH installed.)
IP	Use this option when the remote node is neither of the above (example a Transcoding server, Quality check server)
Aspera Orchestrator	Use this option when the remote node is also an Aspera Orchestrator server.

HOW TO CREATE A REMOTE NODE

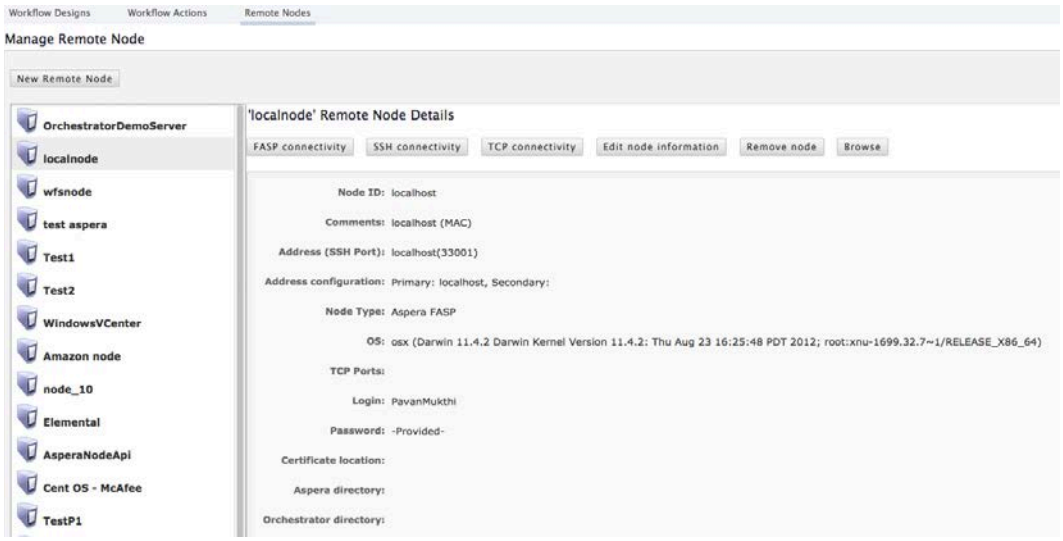
Navigate to Workflows tab. Click the Remote Nodes secondary tab. Click the New Remote Node button to find a popup requesting the information about the node you want to create.



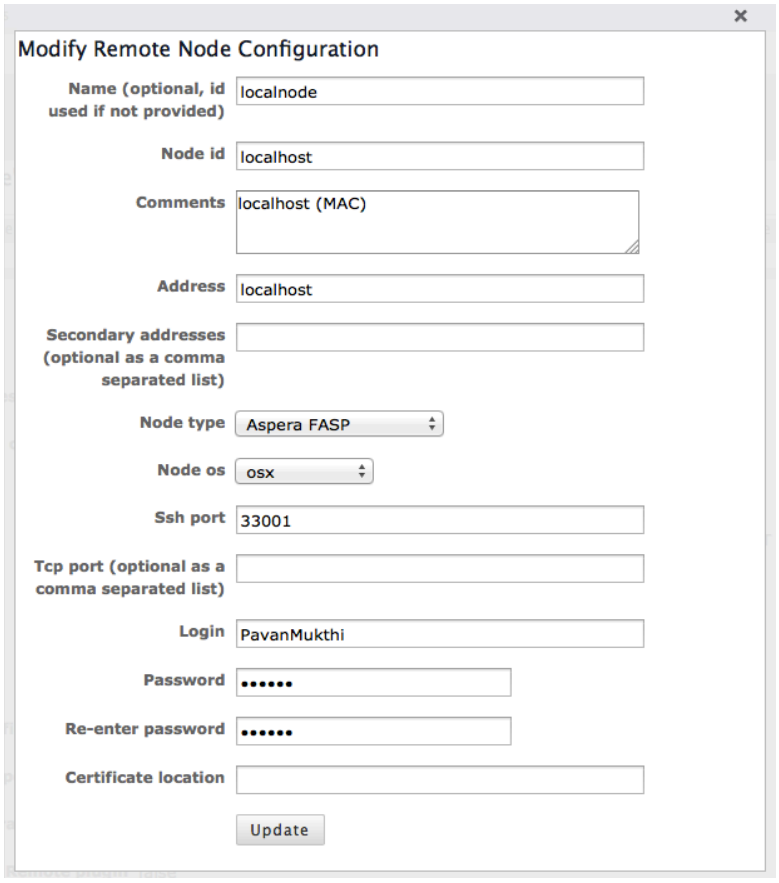


HOW TO EDIT A REMOTE NODE

After creating a remote node, just hover over the created nodes in the left pane until the node name becomes grey. Click it to find the details populated in the right pane.

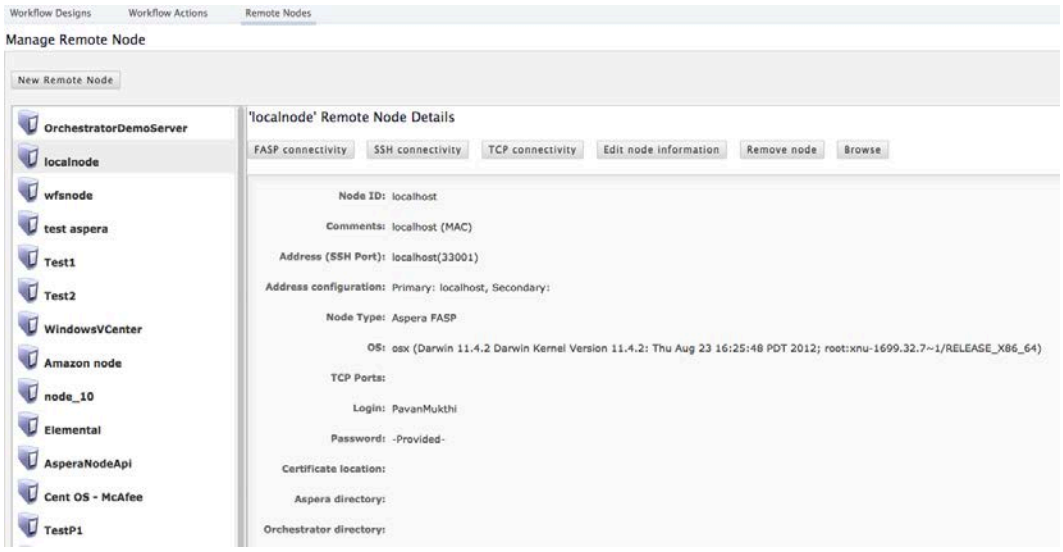


Click Edit node information button to find a popup with the existing information. Make your changes and click Update button located at the bottom of the popup for the new information to be saved.



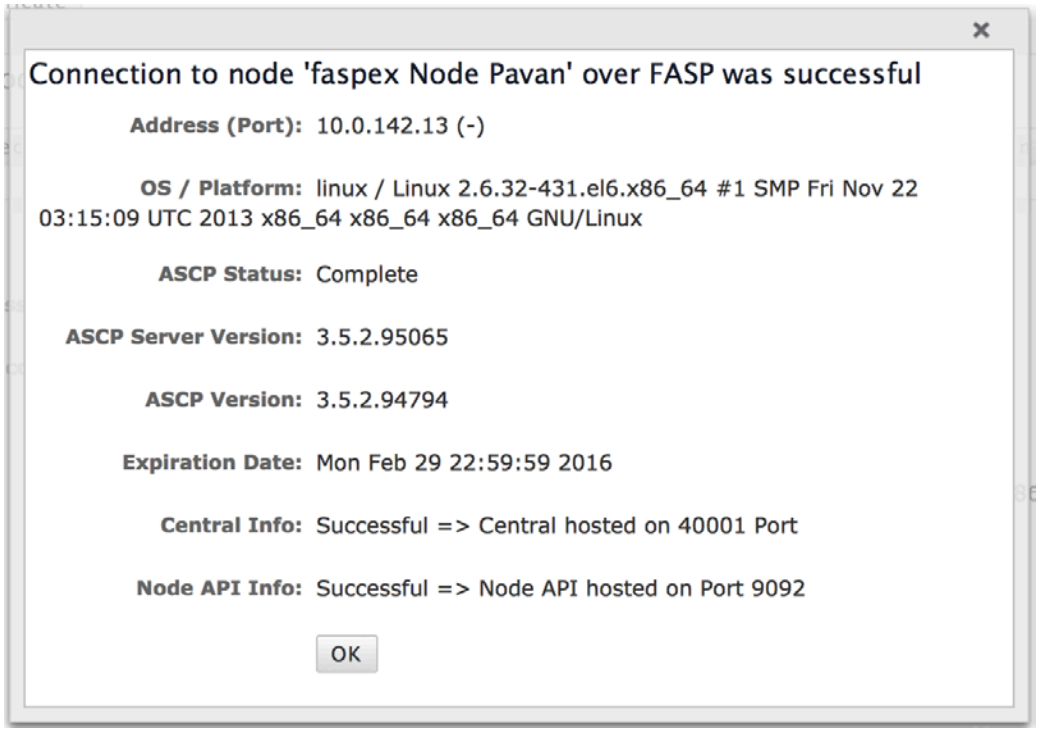
HOW TO TEST A REMOTE NODE

Orchestrator provides functionality to immediately test your remote node settings once created. After creating a remote node, just hover over the created nodes in the left pane until the node name becomes grey. Click it to find the remote node details populated in the right pane.

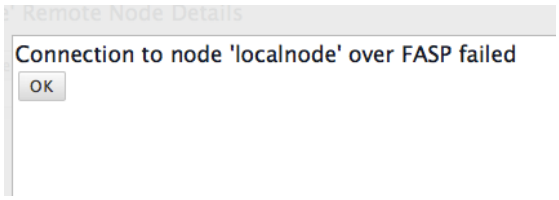


The FASP Connectivity button is displayed only if the node type is "Aspera FASP". When clicked, a FASP connectivity test includes a FASP connection test and an authentication test on the FASP port.

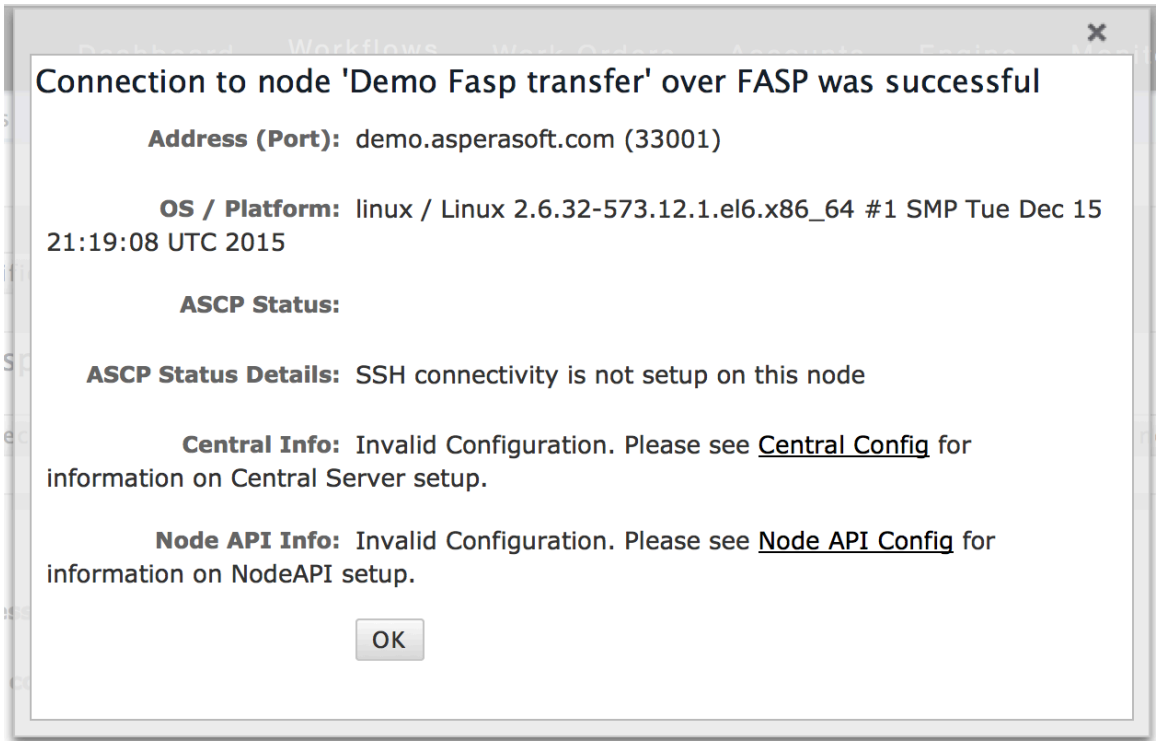
A successful test results in a popup similar to this.



A Failure results in a popup similar to this.



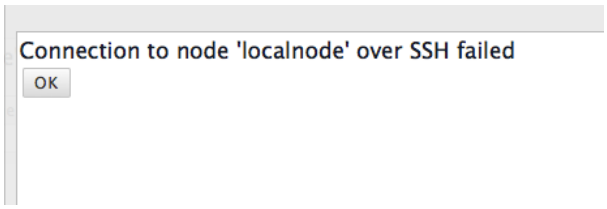
Partial failures or an improper configuration shows as below.



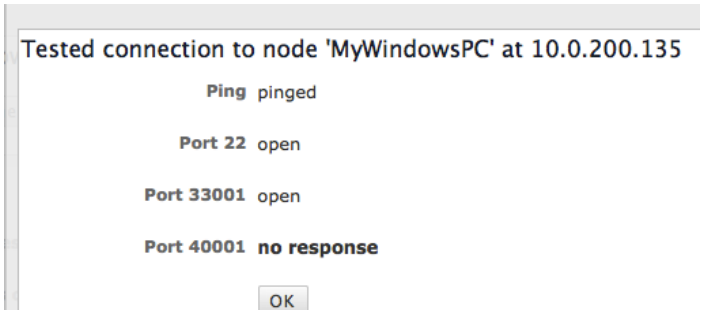
The SSH Connectivity button is displayed if the node type is "Aspera FASP" or "SSH". When clicked, an SSH connection test includes an SSH authentication on the SSH port provided.

A successful test results in a popup similar to this.

A Failure results in a popup similar to this.

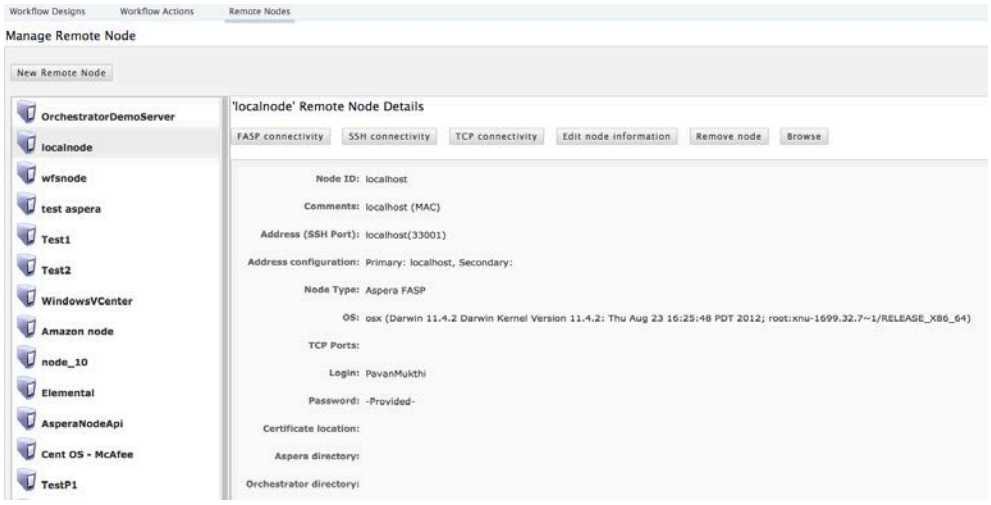


The TCP Connectivity button is displayed for all nodes and when clicked an attempt is made to open a TCP socket. This results in a popup similar to the one below. Open ports and un-opened ports are displayed.

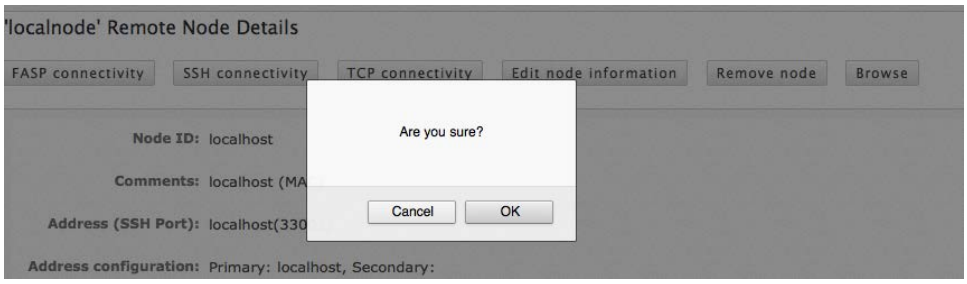


HOW TO REMOVE A REMOTE NODE

After creating a remote node, just hover over the created nodes in the left pane until the node name becomes grey. Click it to find the details populated in the right pane.



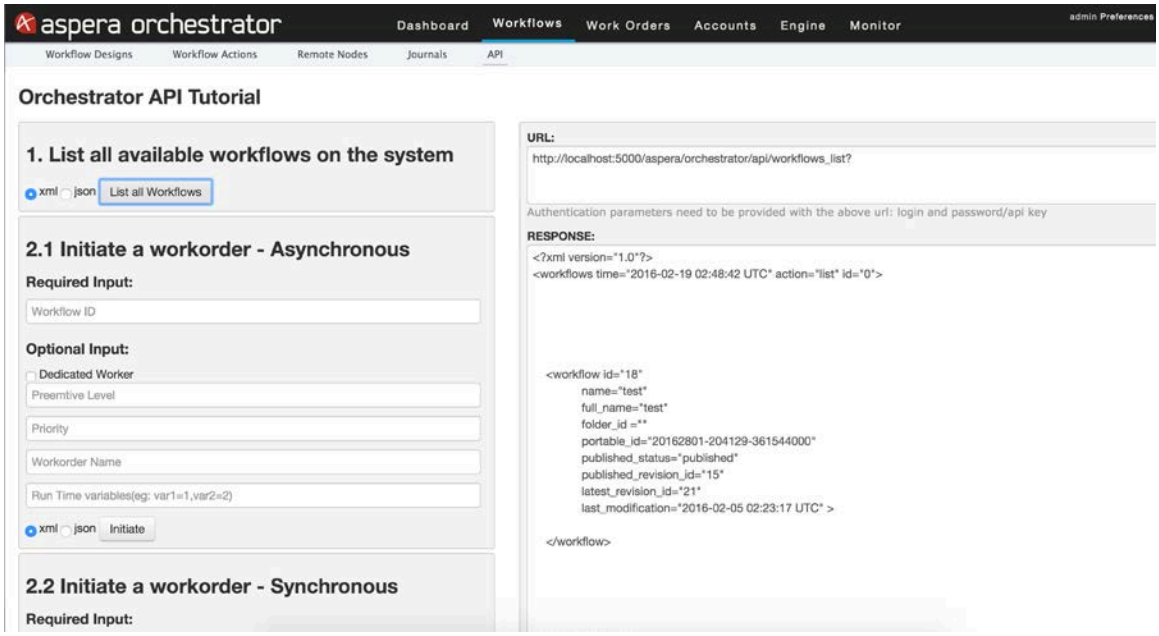
Click the Remove node button to delete the remote node. Confirm your choice in the ensuing popup.



API

HOW TO USE API TUTORIAL

The API tutorial page provides a GUI to access all the APIs exposed by the Orchestrator. It also provides the URL to be used to get the response shown on the page.



As shown above, the page is divided into panes:

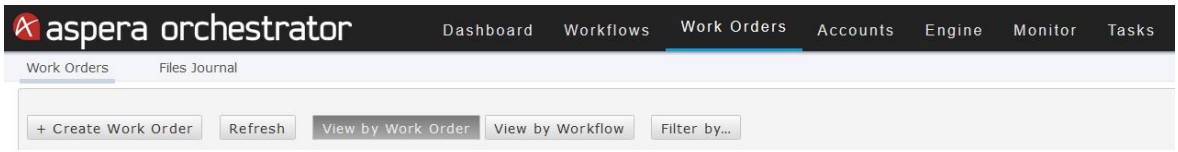
1. The left pane consists of the list of APIs exposed by the Orchestrator.
2. The right pane gives:
   a. The response obtained from the API call made
   b. The URL that should be used to obtain the response.

The URL generated can be used to make the same API call from the browser.

WORK ORDER

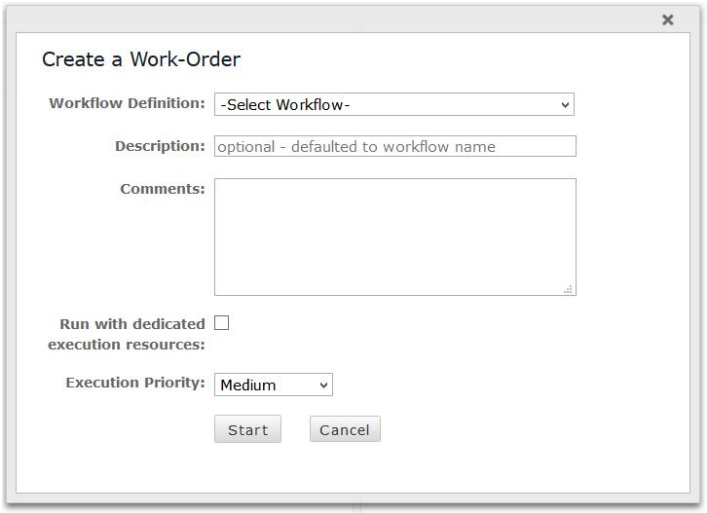
HOW TO CREATE A WORK ORDER

Navigate to the Work Orders tab



Click on Create Work Order

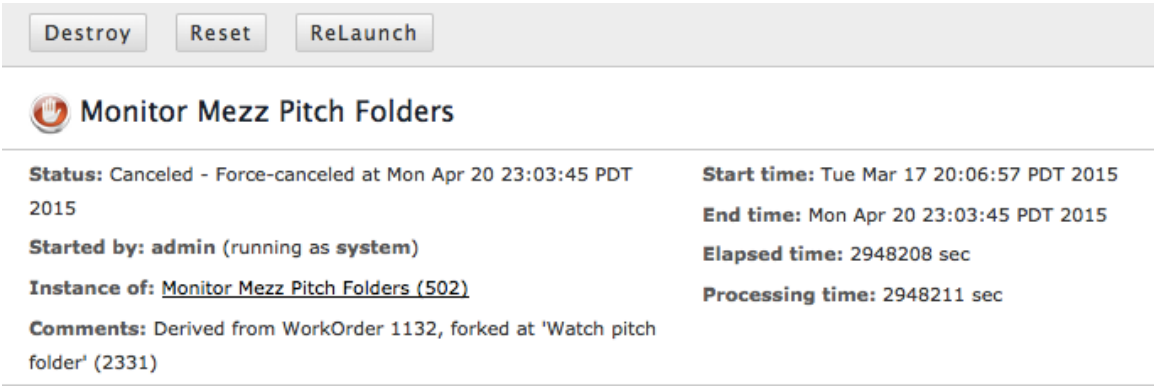
In the popup that appears select the published workflow from the drop-down; enter any description and/or comments for user's understanding; select dedicated resources and priority if necessary. Click start to begin execution of steps of the workflow. Enter run-time parameter values if required.



HOW TO CANCEL A WORK ORDER

Once a Work Order has been initiated it can be cancelled to stop it execute. Select "Work Orders", later select "View by Work Order" sub menu to view all the work orders.

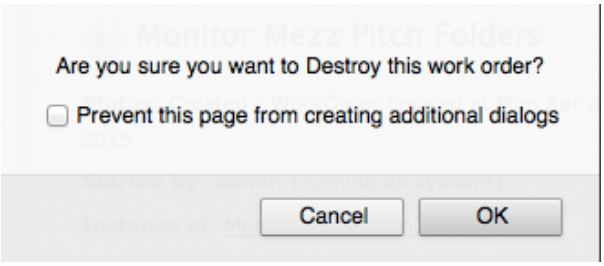
Later click on "Cancel" this will stop execution of the work order and give an option to re-launch the work order.



HOW TO DESTROY A WORK ORDER

Select “Work Orders”, later select “View by Work Order” sub menu to view all the work orders.

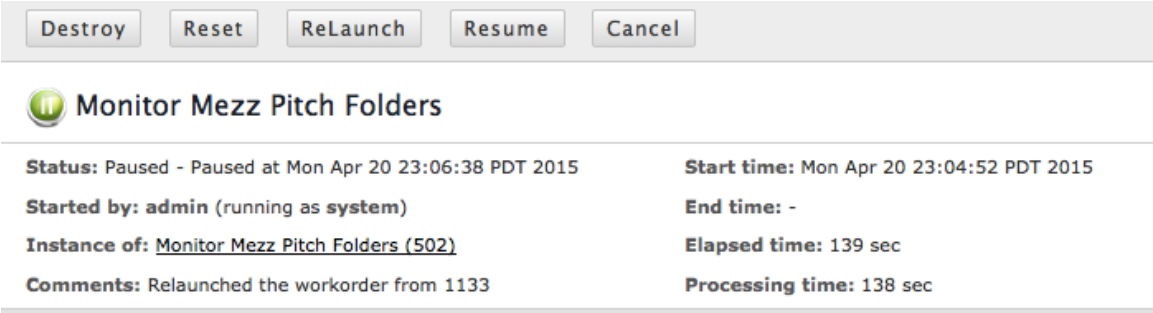
Later click on “Destroy” this will destroy the work order, but before destroying a dialogue box opens ups asking for confirmation from the user.



HOW TO PAUSE A WORK ORDER

Select “Work Orders”, later select “View by Work Order” sub menu to view all the work orders.

Later click on “Pause” this will pause execution of the work order and give an option to resume the work order.



HOW TO RESET A WORK ORDER

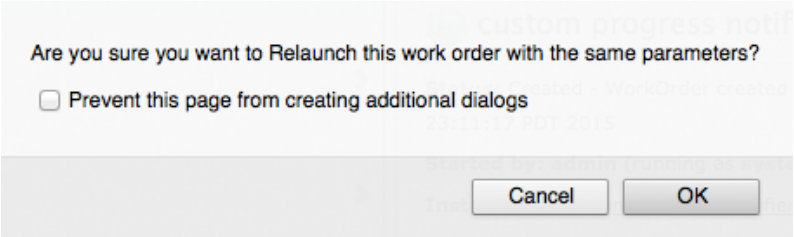
Select “Work Orders”, later select “View by Work Order” sub menu to view all the work orders.

Later click on “Reset” this will reset the work order, but before resetting the work order a dialogue box opens ups asking for confirmation from the user.

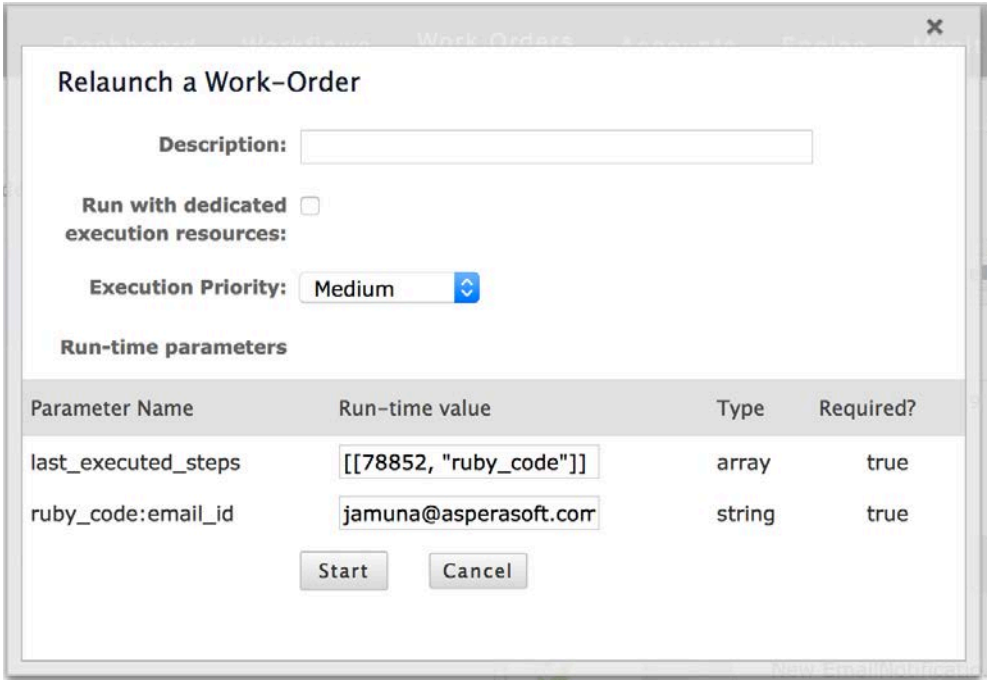
HOW TO RELAUNCH A WORK ORDER

Select “Work Orders”, later select “View by Work Order” sub menu to view all the work orders.

Later click on “Re-launch” this will reset the work order, but before re-launching the work order a dialogue box opens ups asking for confirmation from the user.

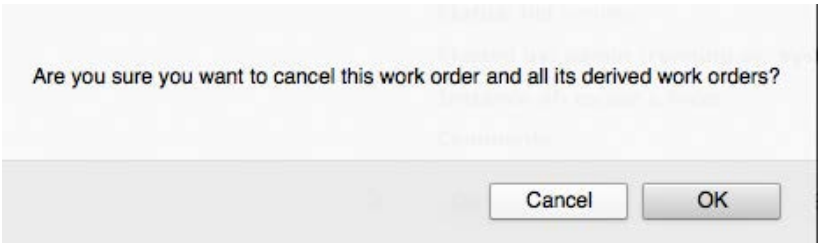


The re-launch window pops-up showing the run-time parameters screen if present as shown below.



HOW TO DESTROY ALL WORK ORDERS

Select “Work Orders”, later select “View by Work Order” sub menu to view all the work orders. “Destroy All” option presents itself when one or more work orders are running. Clicking “Destroy All” button destroys the work orders but first asks for confirmation via the below dialogue box



HOW TO PAUSE ALL WORK ORDERS

Select “Work Orders”, later select “View by Work Order” sub menu to view all the work orders.

“Pause All” option presents itself when one or more work orders are running. Clicking “Pause All” button pauses the work orders and presents a “resume all” button to resume work orders.

HOW TO RESUME ALL WORK ORDERS

Select “Work Orders”, later select “View by Work Order” sub menu to view all the work orders.

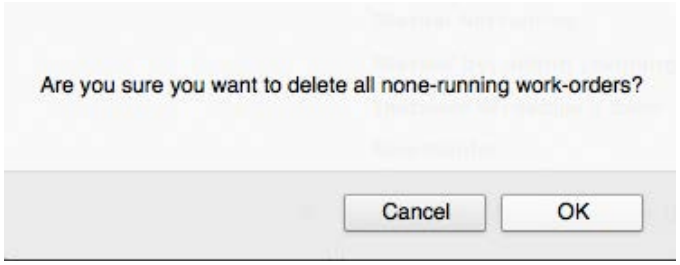
“Resume All” option presents itself when one or more work orders are paused. Clicking “Rseume All” button resumes the work orders and presents a “pause all” button to pause work orders.

HOW TO CLEAN UP WORK ORDERS

Select “Work Orders”, later select “View by Work Order” sub menu to view all the work orders.

“Cleanup All” option presents itself when one or more work orders are paused. Clicking “Cleanup All” button cleans up all work orders.

Select “Work Orders”, later select “View by Work Order” sub menu to view all the work orders. Clean up option presents itself when one or more work orders are running. Clicking “clean up” button cleans up the work orders but first asks for confirmation via the below dialogue box



HOW TO BULK RESTART WORK ORDERS

Bulk Restart is used to restart multiple work-orders based on the status of the step/ work-order at once. Select “Work Orders”, later select “View by Work Order” / “View by Workflow” sub menu to view all the work orders.

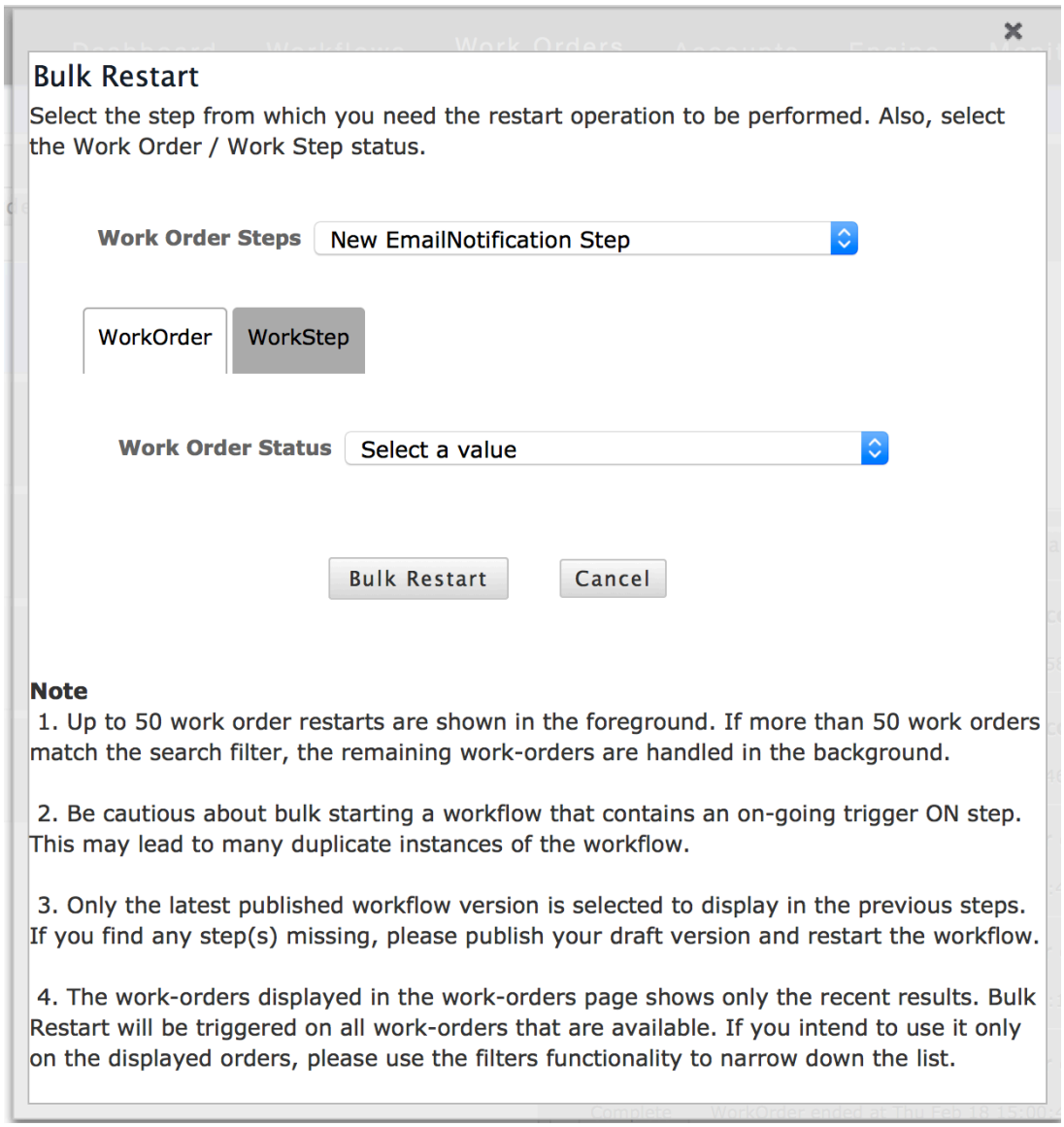
“Bulk-Restart” option presents itself when more than 1 work orders are present together. Clicking this button restarts all work orders.

Restart can be performed based on a

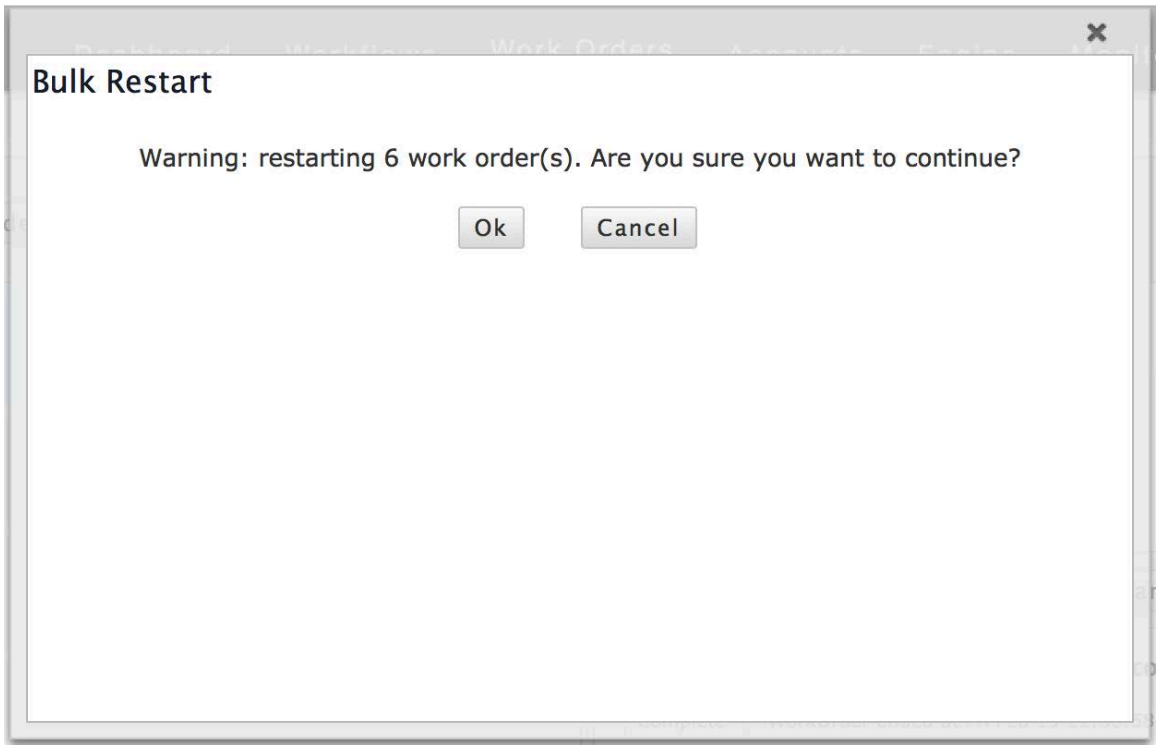
- WorkStep and WorkOrder Status filter (or)
- WorkStep and WorkStep status filter.

With the former it picks all the WorkOrders with the Status provided and restarts from the WorkStep Provided. For the later, it picks all the WorkOrders with The given WorkStep in the given WorkStep Status and restarts them from the WorkStep action.

Bulk-Restart is handled both by the UI process and the backend Worker process to not crash the UI process with excessive load.



On entering, it shows the below confirmation page.

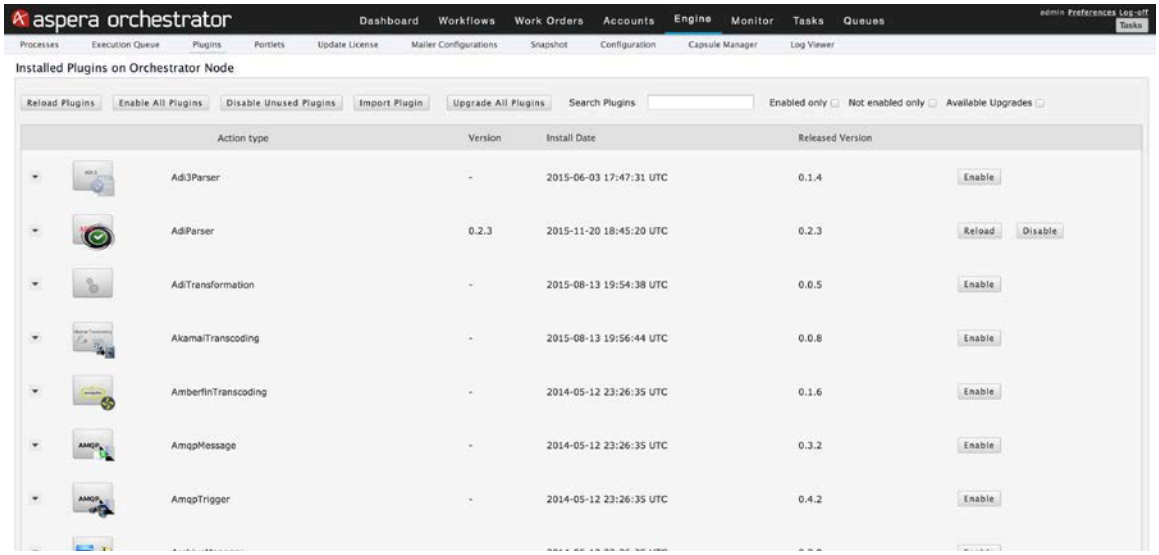

Bulk Restart
Warning: restarting 6 work order(s). Are you sure you want to continue?
Ok
Cancel

PLUGINS

Plugin Manager screen found under Engine -> Plugins has tools to enable/disable plugins used in Orchestrator workflows.

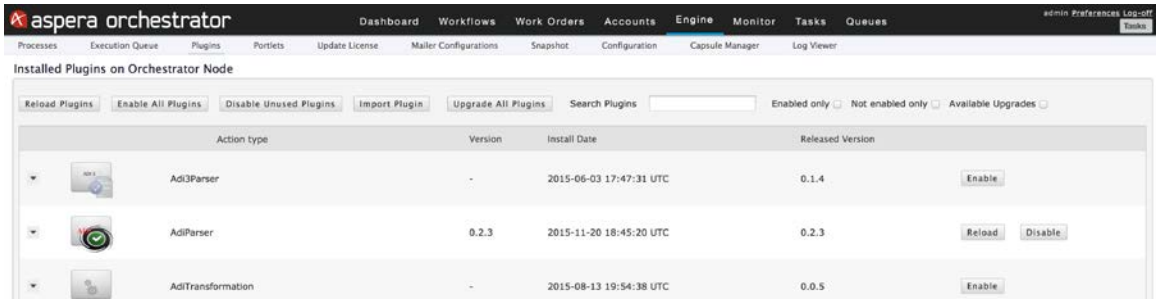
HOW TO SEARCH FOR A PLUGIN

Select the Engine menu, and then click the Plugins tab. In the Search Plugins box, enter part of the name of the plugin you are looking for. The list will update with the plugin that matched the search string.



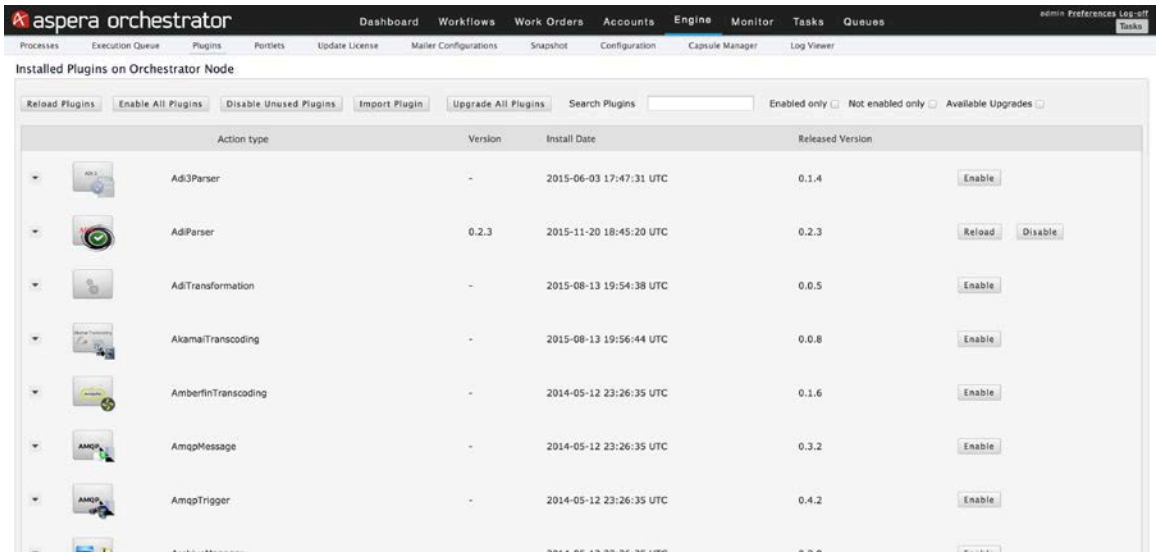
HOW TO ENABLE A PLUGIN

Search for the plugin as described in the previous section, or select the Engine menu, click the Plugins menu, and then scroll to the plugin you wish to enable. Click the Enable button for the desired plugin. You can also select the “Not enabled only” box. This will only show the plugins that are not already enabled. This makes it easier to scroll and select the desired plugin.



HOW TO DISABLE A PLUGIN

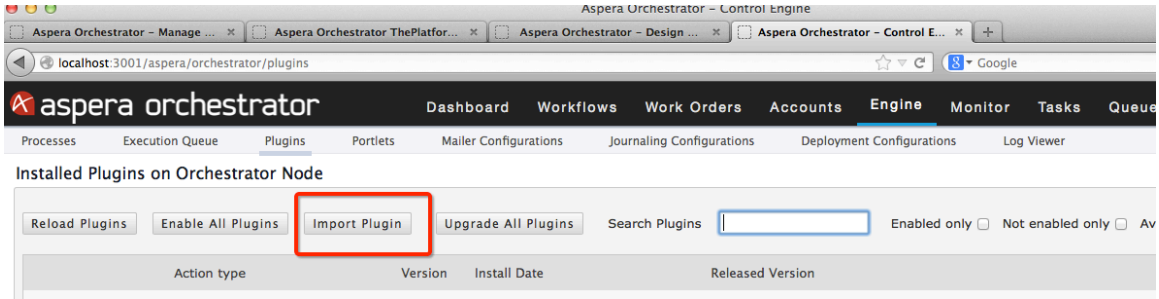
Search for the plugin as described in the previous section, or select the Engine menu, click the Plugins menu, and then scroll to the plugin you wish to disable. Click the 'Disable' button for the desired plugin. You can also select the 'Enabled only' box. This will only show the plugins that are already enabled. This makes it easier to scroll and select the desired plugin.



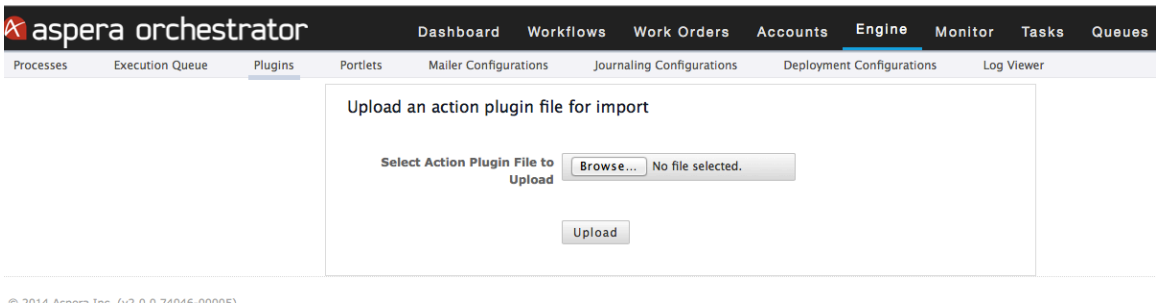
HOW TO IMPORT A PLUGIN

Importing a plugin, will enable a that plugin inside Orchestrator's plugin repository. If that plugin was already enabled, that plugin (code, version, functionality) will be overwritten by the plugin being imported. If no instance of the plugin is found (usually happens with newly developed plugins) the plugin being imported will create its own entry in the plugin manager.

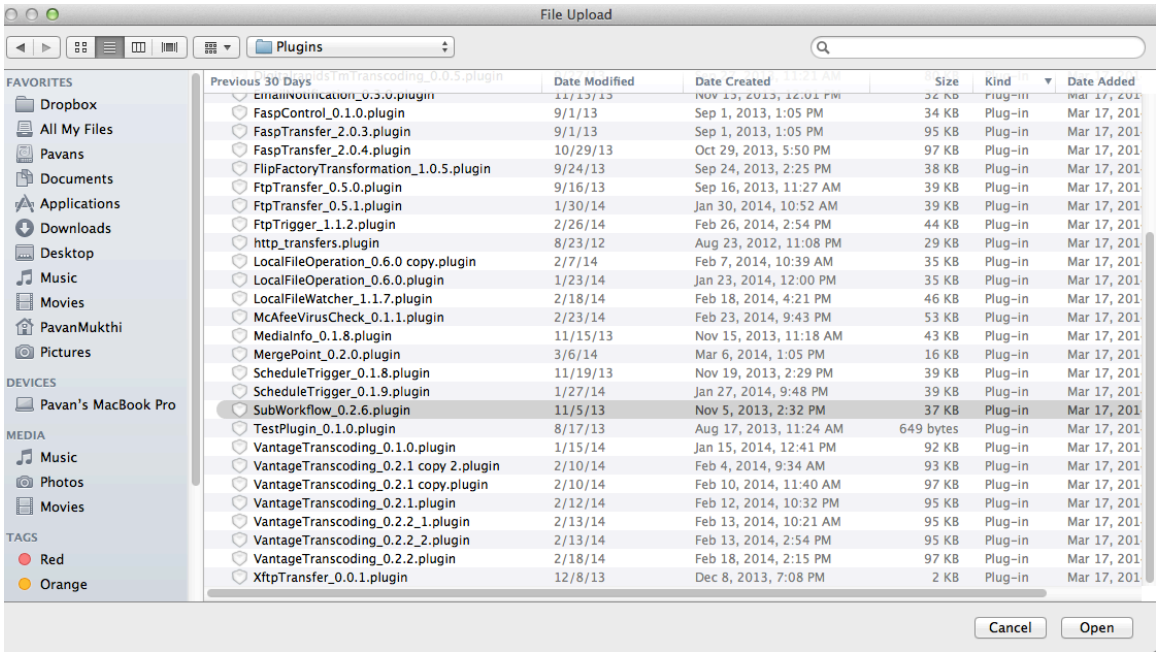
Navigate to Engine tab and click on Plugins in the secondary navigation to find the plugin manager page. In the plugin manager page, find the Import Plugin button.



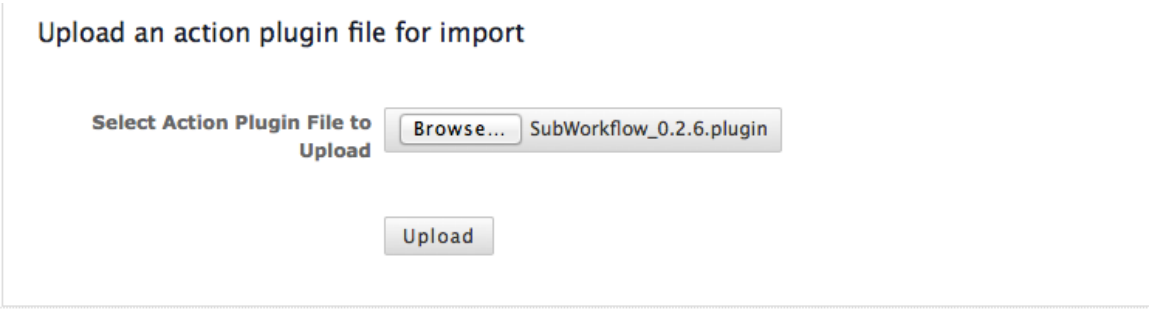
Click the Import Plugin button to find the below page.



Click the Browse button to find the File Browser popup and select a valid plugin to upload. A valid plugin contains a .plugin extension. Contains a version with the least being 0_0_1



Selection of a plugin results in the name being populated beside the Browse button.



Click upload to install the plugin.

The resulting page will be the Plugin manager page with the status of the import. A success message will be displayed when the import is successful. An error message will be displayed when the import is not successful.

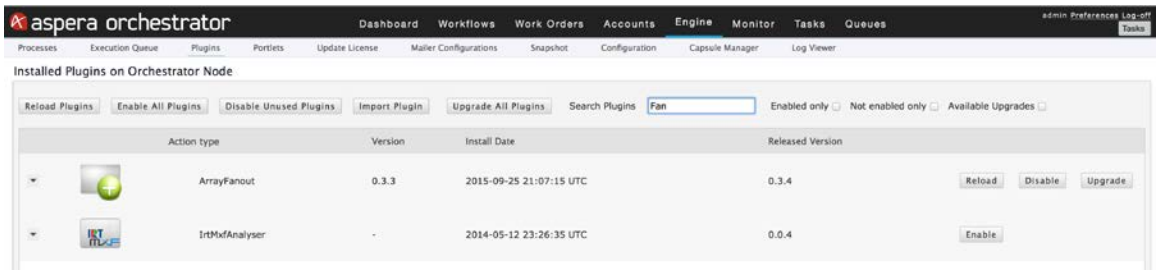
HOW TO RELOAD A PLUGIN

What does a Reload do?

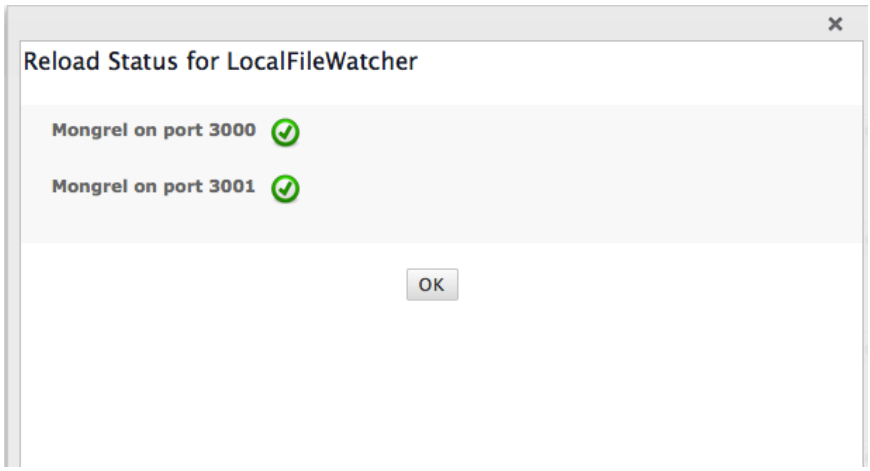
A plugin reload will force Orchestrator to take the newest code into affect. A Reload is always followed by forced maintenance by clicking the "Force Maintenance" on the Process tab under Engine.

Select the Engine menu, and then click the Plugins tab. Either use Search Plugins box, enter part of the name of the plugin you are looking for or scroll down to find the plugin you want to Reload.

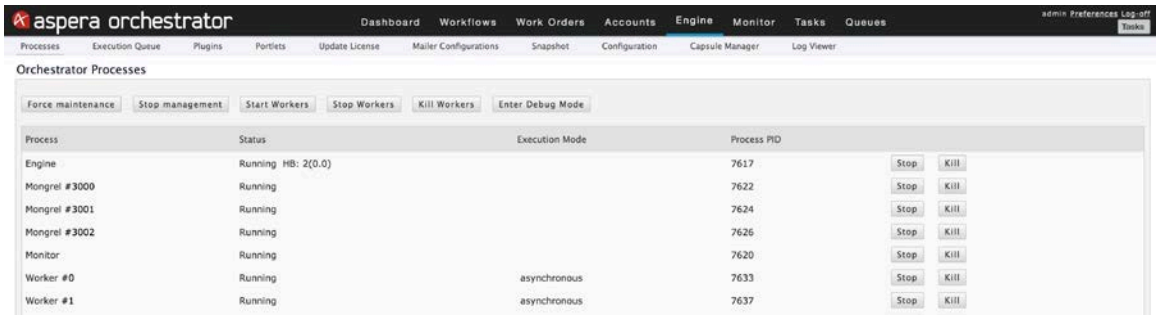
Click Reload button to reload the plugin.



Once clicked, the reload status on the Mongrels is displayed. Click Ok to find the Plugin manager page reload.



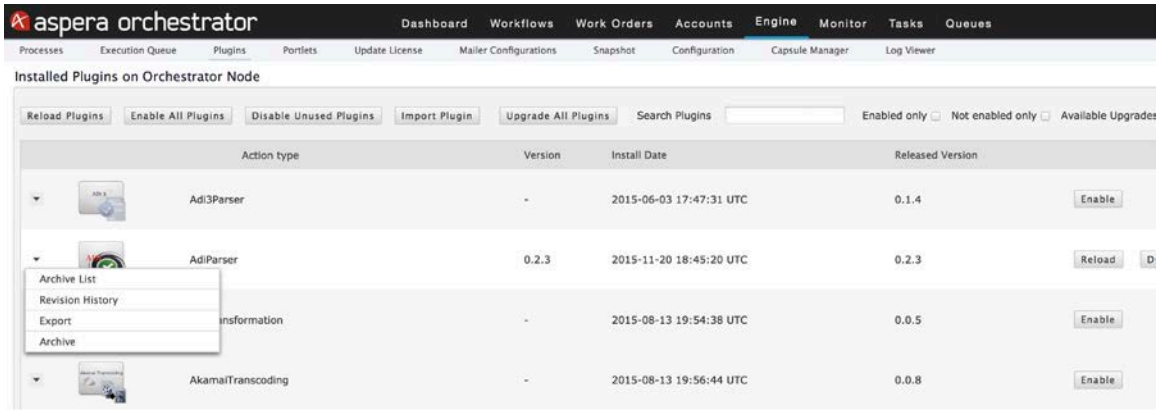
Navigate to Engine -> Processes page and click on the 'Force maintenance' button to complete the plugin Reload.



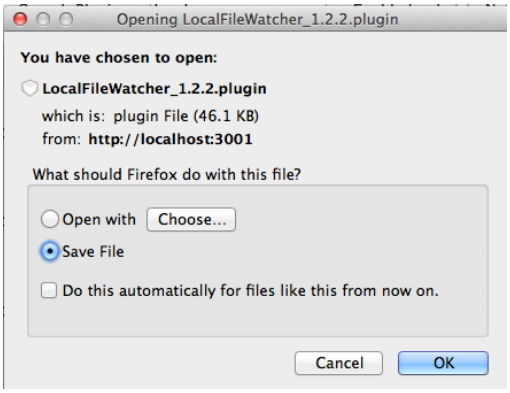
HOW TO EXPORT A PLUGIN

An export of a plugin packages all the plugin files

Select the Engine menu, and then click the Plugins tab. Either use Search Plugins box, enter part of the name of the plugin you are looking for or scroll down to find the plugin you want to export.



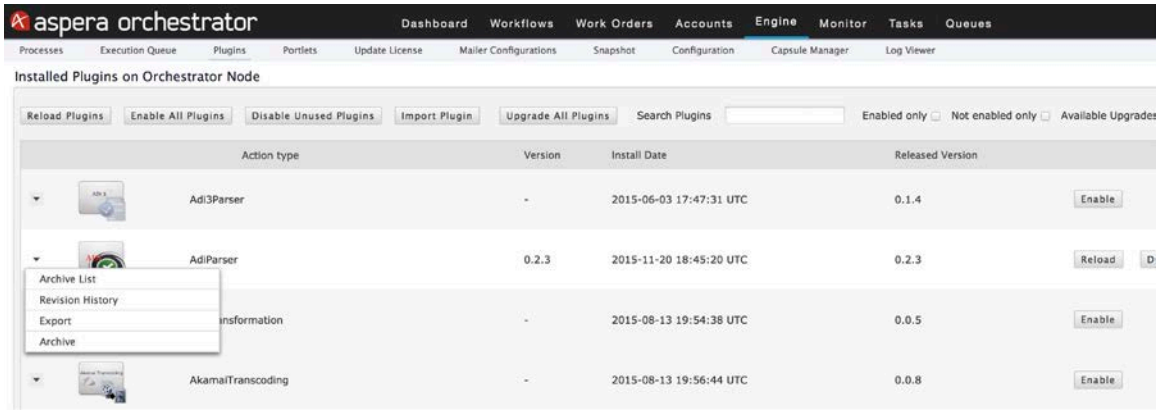
Click Export link on the plugin menu (down arrow button on the left of plugin icon) to export the plugin. Sometimes, browsers will ask you to confirm saving the file on your file system. Click Ok to continue the export.



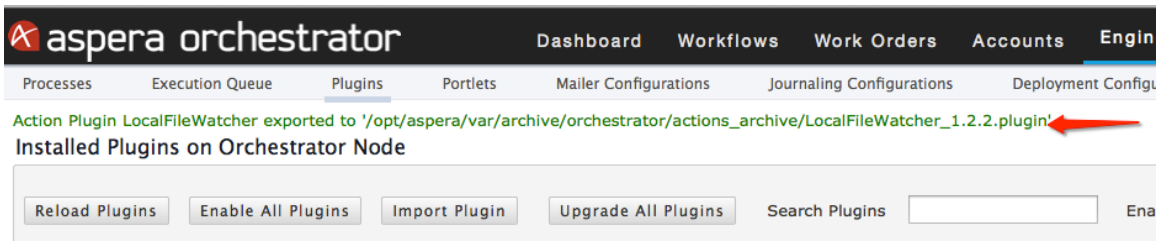
HOW TO ARCHIVE A PLUGIN

An archive saves a copy of the plugin into the "actions archive" directory Orchestrator's archive directory. Refer to the Orchestrator install directories section at the top to find where your archive directory is.

Select the Engine menu, and then click the Plugins tab. Either use Search Plugins box, enter part of the name of the plugin you are looking for or scroll down to find the plugin you want to archive.

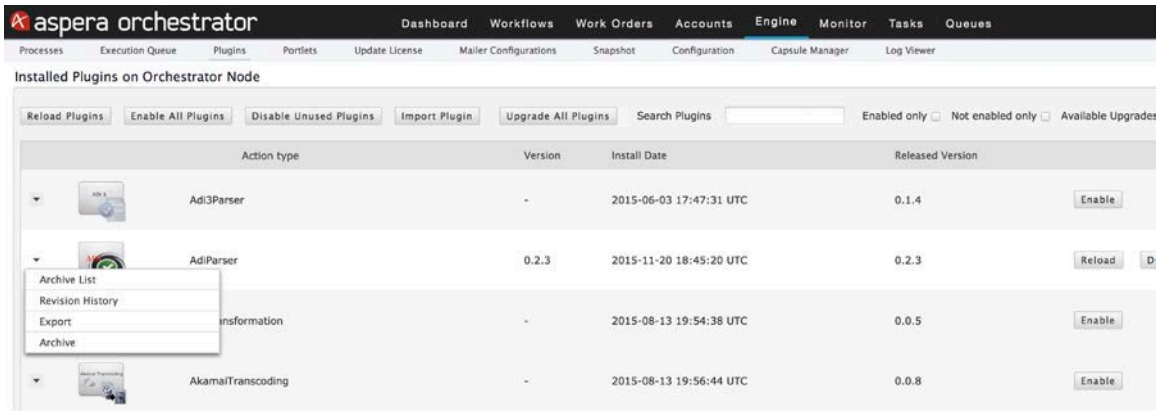


Click the Archive link on the menu (down arrow button on the left of plugin icon) to archive the plugin. The page reloads with a success message (if the archive is successful).

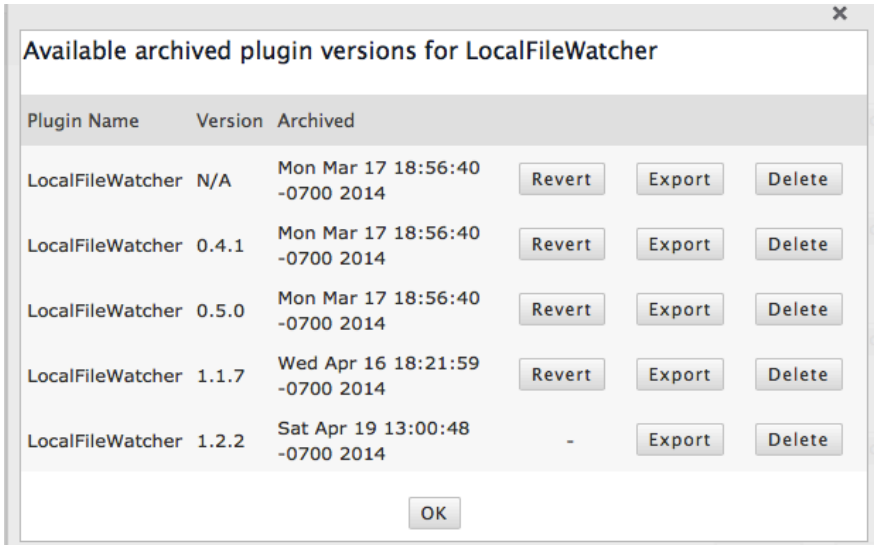


HOW TO SEE THE ARCHIVE LIST

For each plugin, Orchestrator maintains an archive list. Select the Engine menu, and then click the Plugins tab. Either use Search Plugins box, enter part of the name of the plugin you are looking for or scroll down to find the plugin you want to see the archive list of.



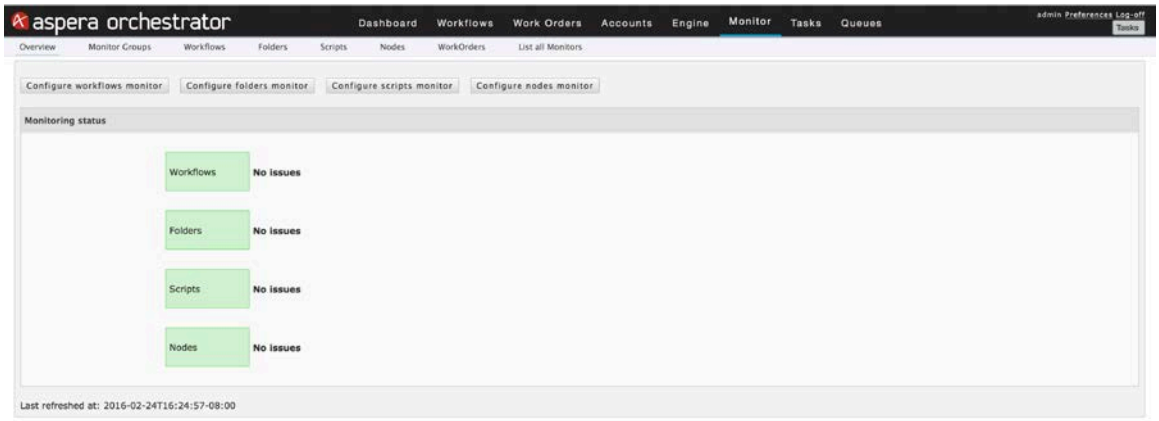
Click Archive List link on the menu (down arrow button on the left of plugin icon) to find a popup showing you the list of archived plugins.



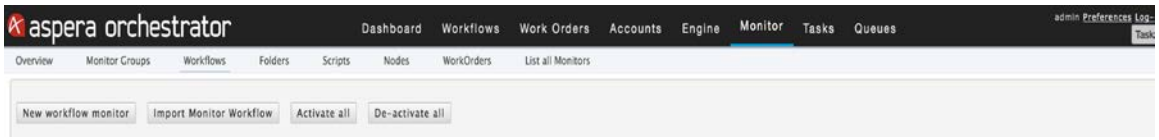
MONITORS

HOW TO ADD A WORKFLOW MONITOR

Select the Monitor menu, and then click the "Configure workflows monitor" button. This will take you to the screen where you can add your new workflow monitor.



Click the "New workflow monitor" button.

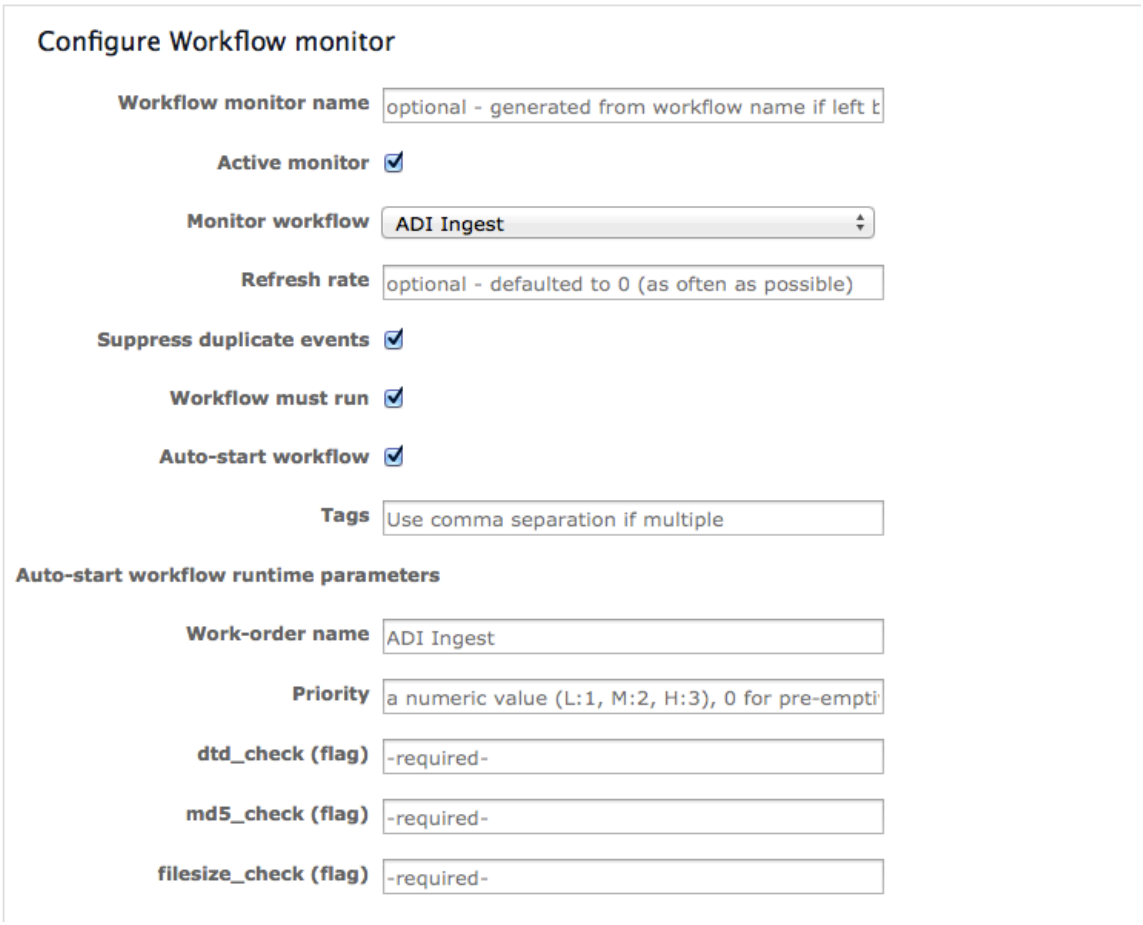


That takes you to the screen where you can define your new monitor. In the following example, a monitor is created for "ADI Ingest" workflow and the monitor is active, and it must run, and if a work order stops, will be automatically started.

Runtime parameters can be provided to the work order via the monitor.

Options described

Option	Description
Active Monitor	The Monitor process will add the new monitor to its list of workflows to monitor
Suppress duplicate events	If a monitor state doesn't change for a while, this flag stops a notification being generated at every check.
Workflow must run	When checked, the monitor overview changes to red if no work order is found in running state
Auto-start workflow	When checked, the monitor starts a new work order if no work order is found in running state
Tags	Comma separated entries can be added to tags to search and differentiate work orders started by monitors

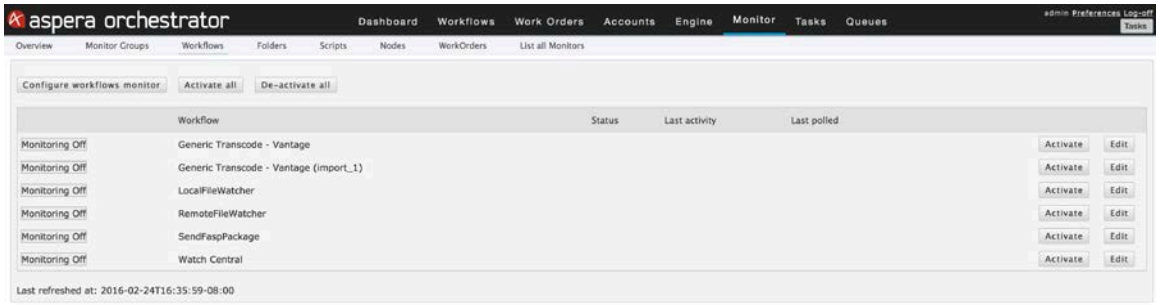

Configure Workflow monitor
Workflow monitor name
optional - generated from workflow name if left b
Active monitor
Monitor workflow
ADI Ingest
Refresh rate
optional - defaulted to 0 (as often as possible)
Suppress duplicate events
Workflow must run
Auto-start workflow
Tags
Use comma separation if multiple
Auto-start workflow runtime parameters
Work-order name
ADI Ingest
Priority
a numeric value (L:1, M:2, H:3), 0 for pre-empti
dtd_check (flag)
-required-
md5_check (flag)
-required-
filesize_check (flag)
-required-

HOW TO ACTIVATE AND DEACTIVATE WORKFLOW MONITORS IN BULK

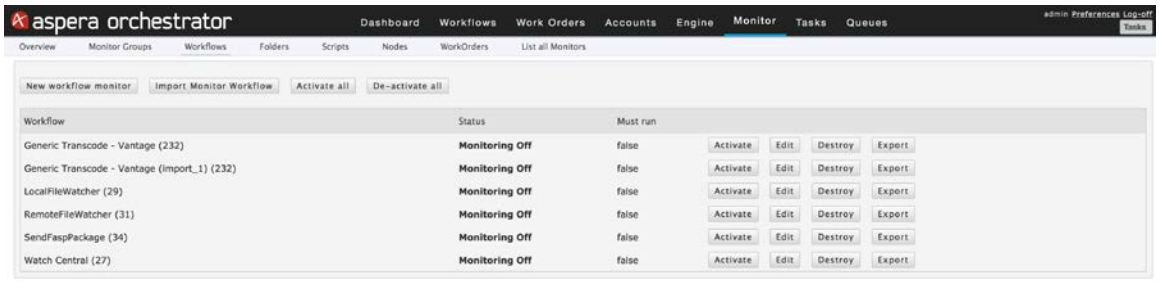
All the workflow monitors can be activated and de-activated providing ease while maintenance and taking snapshots, where user needs to go to each monitor and deactivate individually.

'Activate all' activates all the workflow monitors that are not running or inactive. 'De-activate all' button will deactivate or stop running all the workflow monitors that are running. These buttons can be accessed from the monitor dashboard's page and from the Configure workflow monitor page. Show in below screens.

Workflow monitor dashboard:

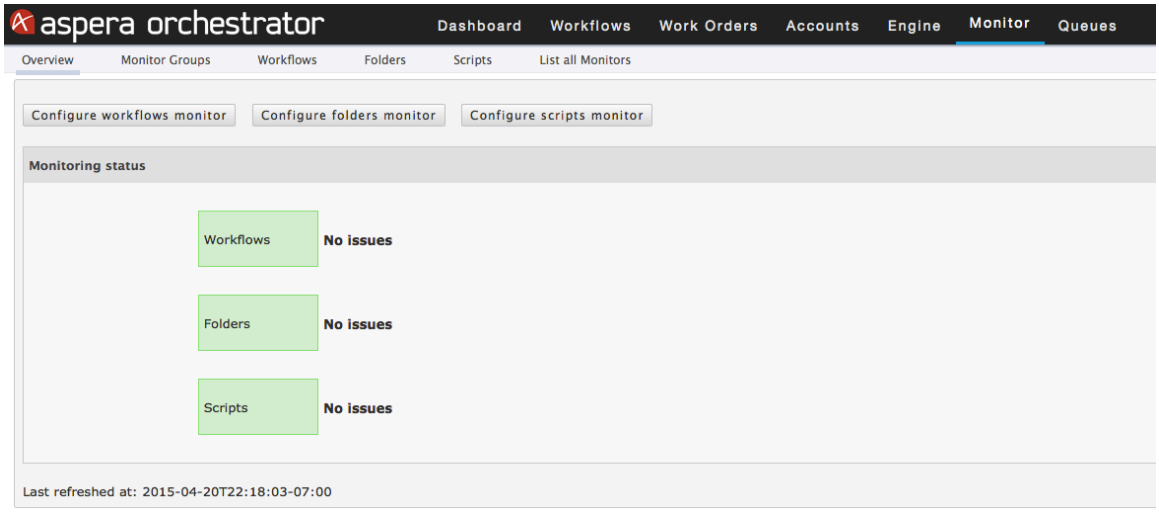


Configure workflow monitor:

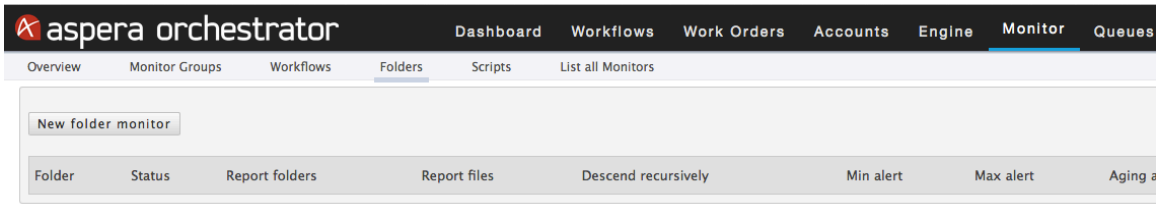


HOW TO ADD A FOLDER MONITOR

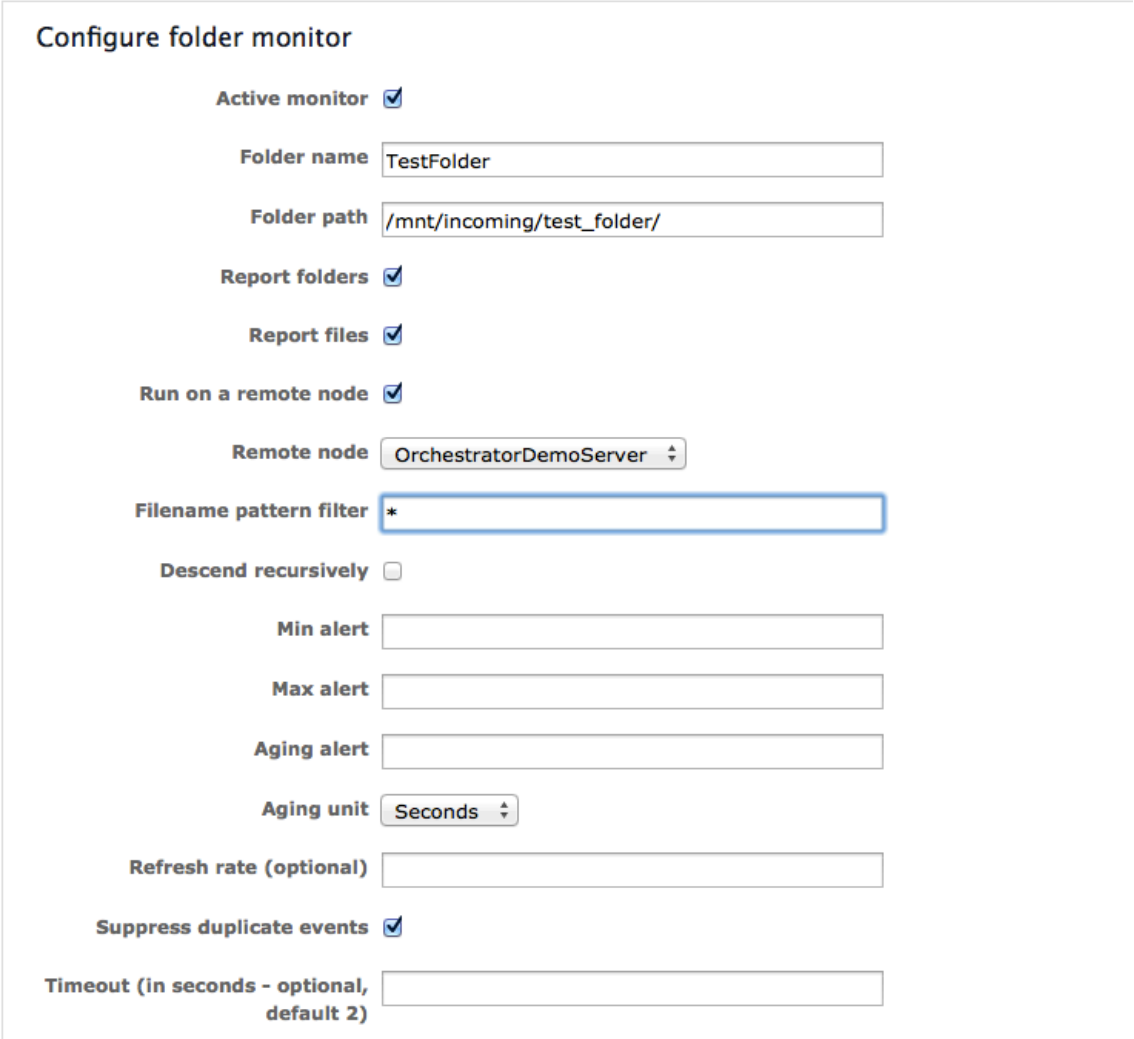
Select the Monitor menu, and then click the "Configure folders monitor" button. This will take you to the screen where you can add your new workflow monitor.



Click the “New folder monitor” button.



This takes you to the screen where you can define your new folder monitor. In the following example, a folder monitor is created for a folder located at /mnt/incoming/test_folder/.

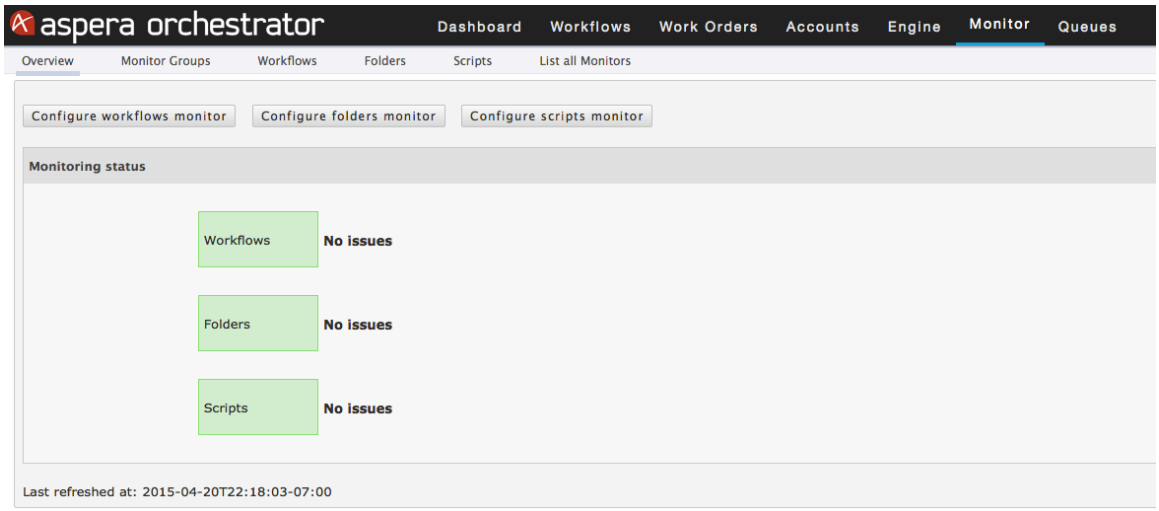


Options described

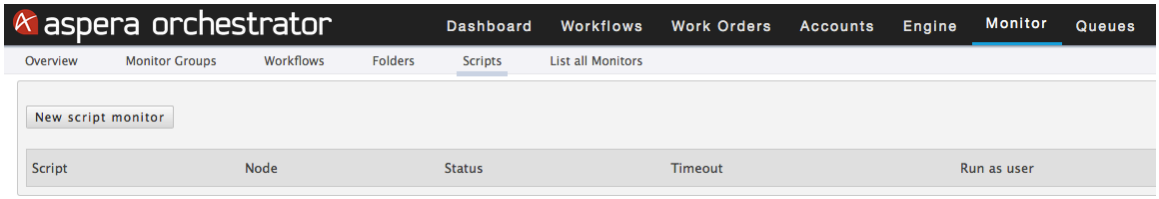
Option	Description
Active Monitor	The Monitor process will add the new folder monitor to its list of folders to monitor
Report folders	Stats include folders inside the folder
Report files	Stats include files inside the folder
Run on a remote node	When checked, the folder monitor starts a SSH tunnel to the remote node and obtains information about the folder. Uncheck if the folder is local to Orchestrator.
Descend recursively	When checked, the stats about the folder will include sub folders and their files
Filename pattern	A glob pattern to include stats about certain types of files. For example, *.mov will only report files with mov extension

HOW TO ADD A SCRIPT MONITOR

Select the Monitor menu, and then click the "Configure scripts monitor" button. This will take you to the screen where you can add your new scripts monitor.



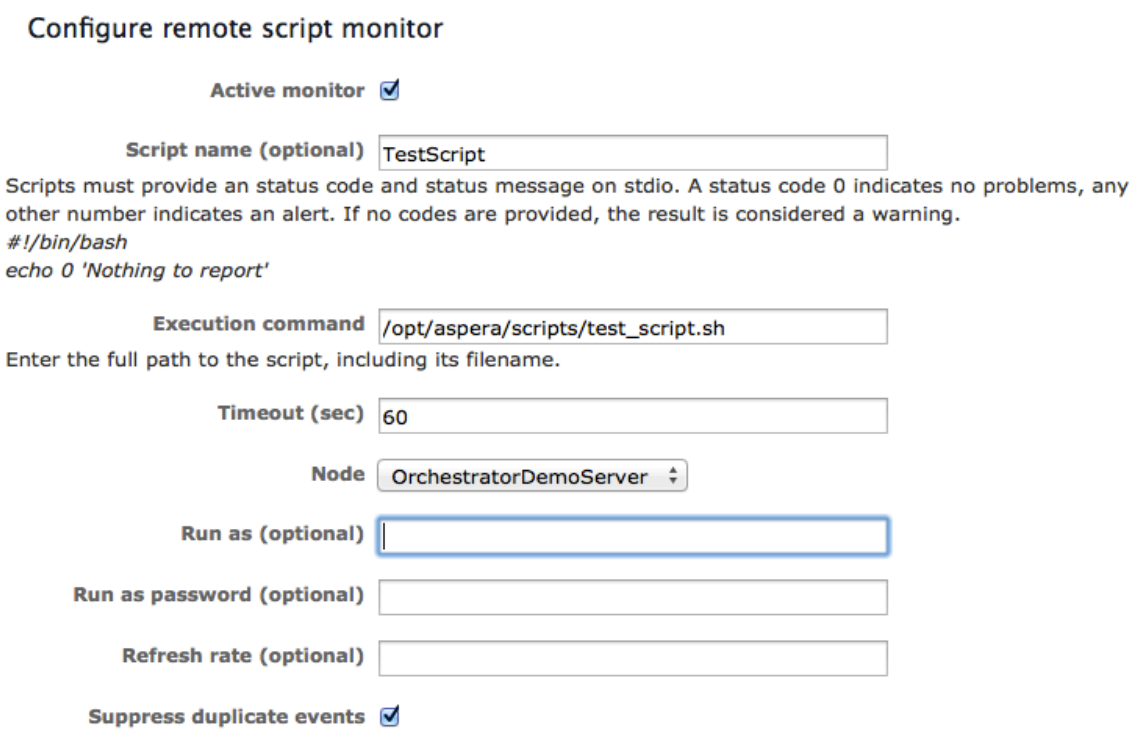
Click the "New script monitor" button.



This takes you to the screen where you can define your new script monitor. In the following example, a script monitor is created for a script located at /opt/aspera/scripts/test_script.sh on a node OrchestratorDemoServer. If the script needs to run locally, create a localhost node.

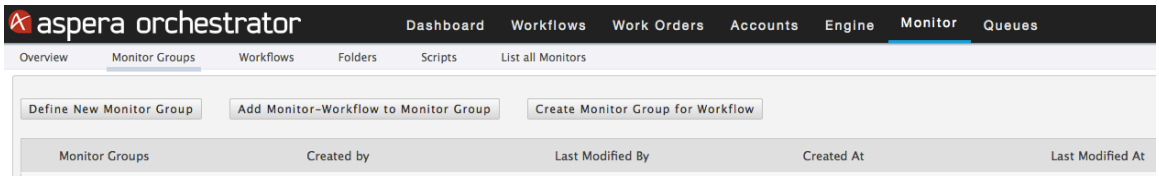
Options described

Option	Description
Timeout	Time in seconds to abort running the script if it does not respond with a valid exit code
Suppress duplicate events	If a monitor state doesn't change for a while, this flag stops a notification being generated at every check.
Run As	Run the script as a user different from the SSH login user
Run As Password	Password of the above user
Refresh rate	Frequency in which the script stats should be updated in the Overview screen



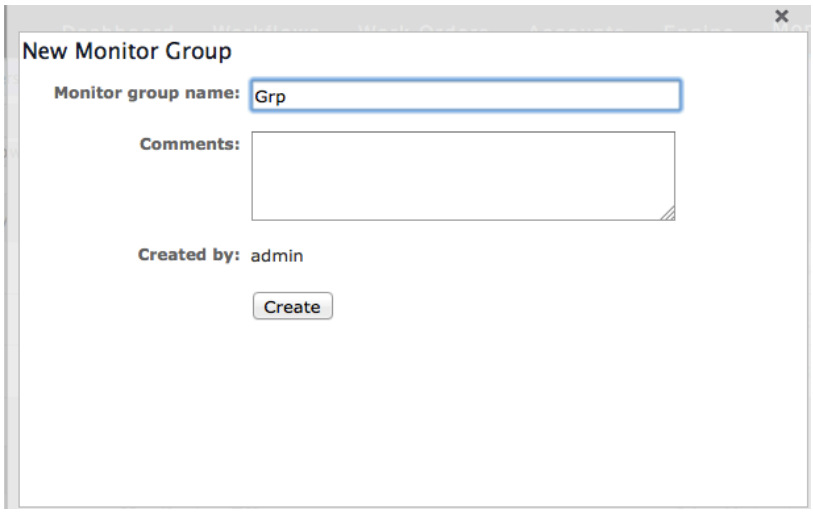
HOW TO GROUP WORKFLOW MONITORS

Select the Monitor menu, and then click the "Monitor Groups" sub menu. This will bring up the Monitor Groups page. A monitor group allows grouping of workflow monitors to perform bulk operations on all the workflow monitors.

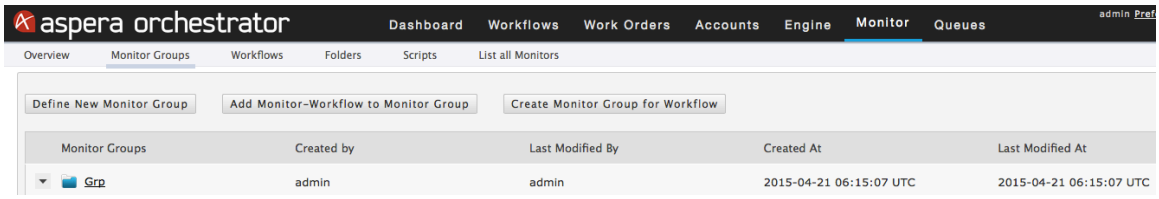


CREATE A NEW MONITOR GROUP

Click on Define New Monitor Group button to get a popup requesting details like name (mandatory) and comments

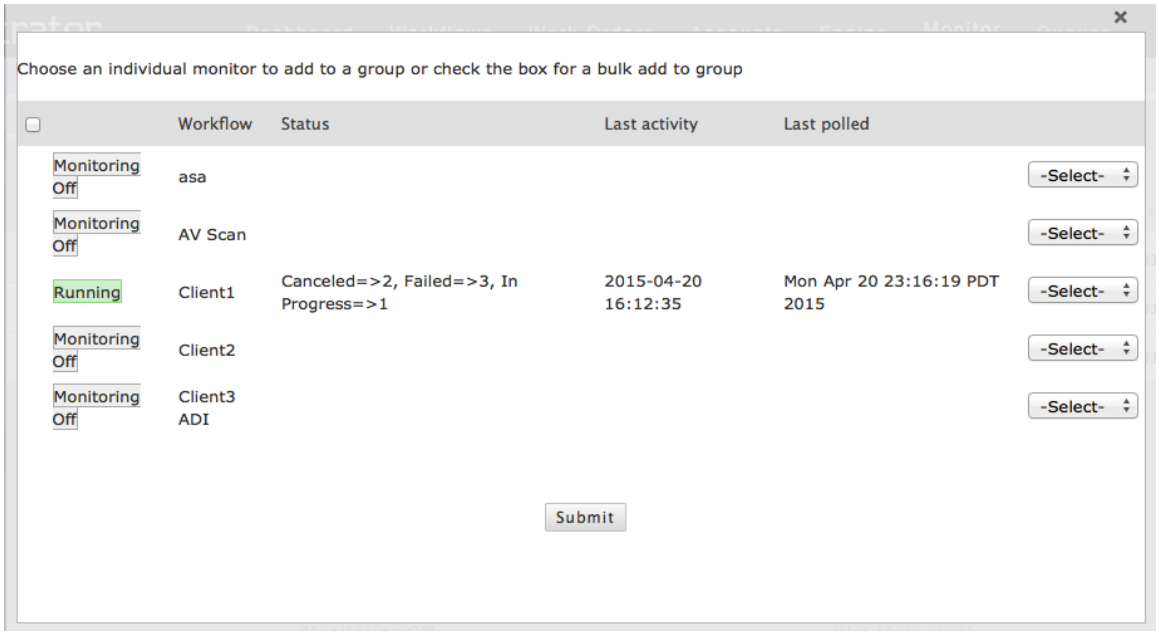


Submit the form by click on Create button. The monitor group is now shown on the page.



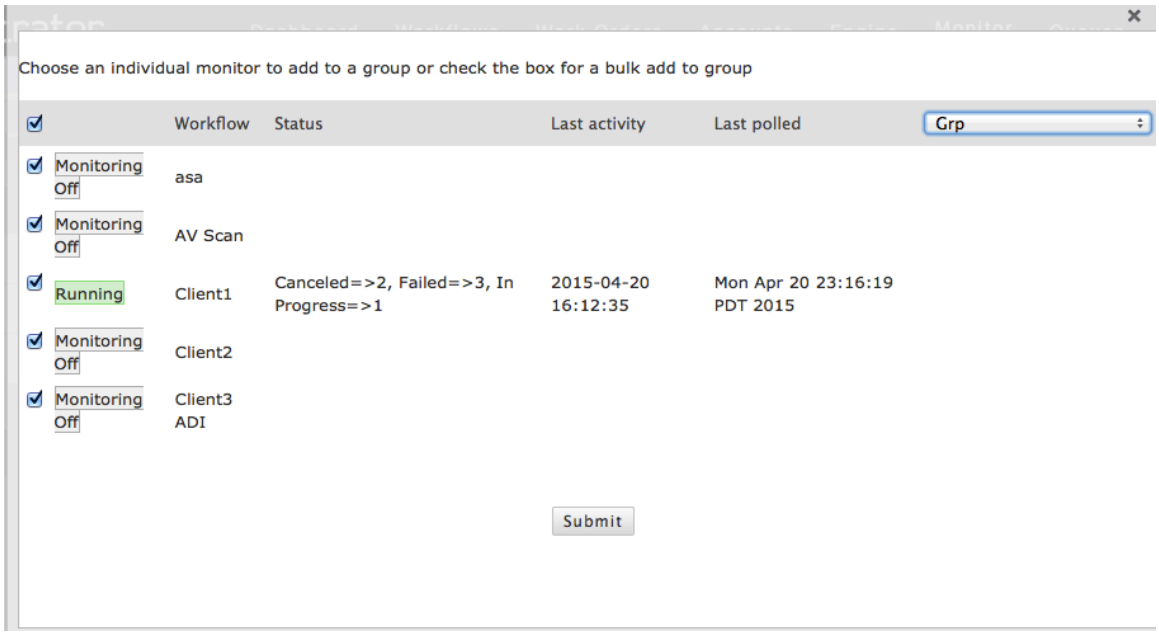
ADD ONE OR MORE MONITOR TO MONITOR GROUP

Click on Add Monitor-Workflow to Monitor Group to get the following pop up

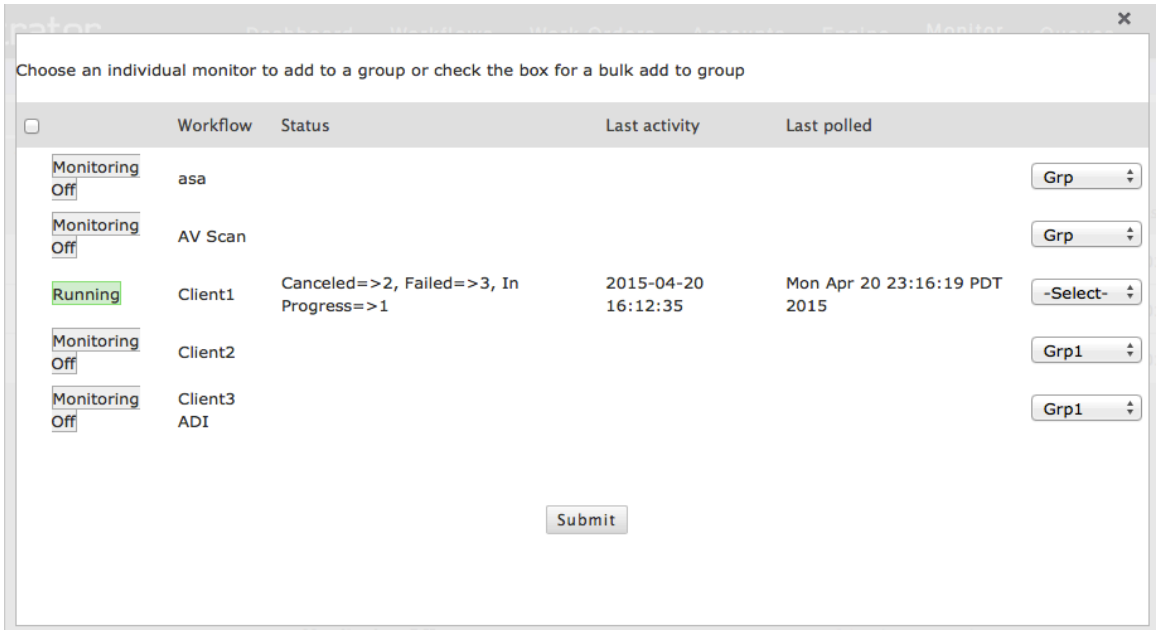


All workflow monitors can be added to one monitor group, or each workflow monitor can be added to different monitor group.

Scenario 1 – All workflow monitors added to Monitor Group “Grp”.

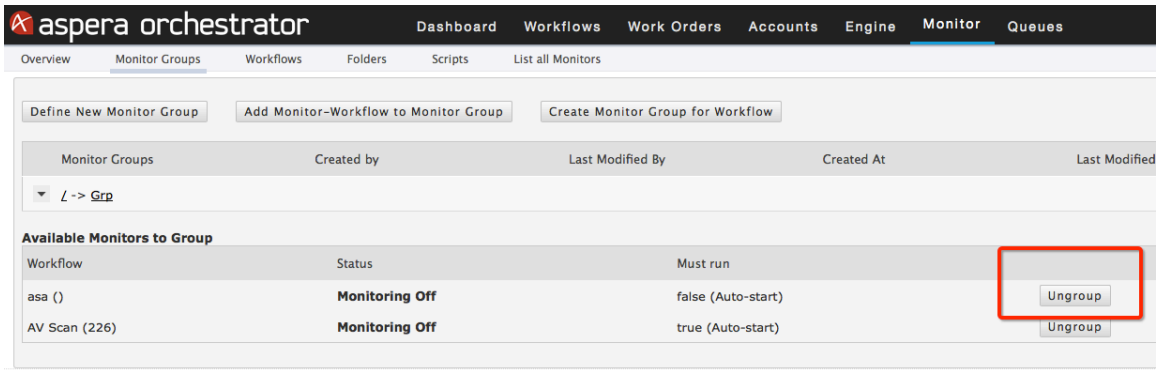


Scenario 2 – Some workflow monitors added to Monitor Group “Grp” and some to Monitor Group “Grp1”



UNGROUP MONITOR FROM MONITOR GROUP

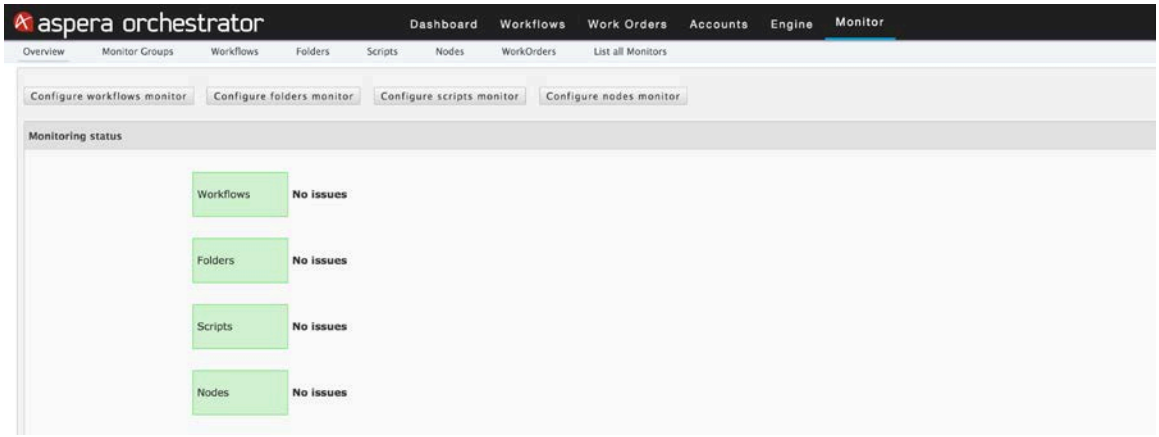
Navigate into the monitor group containing the workflow monitors and click Ungroup button



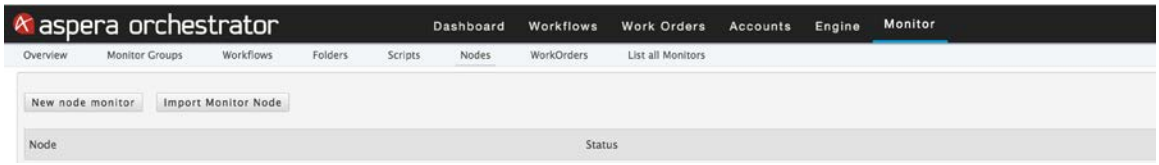
HOW TO ADD A REMOTE NODE MONITOR

Remote Node Monitor allows to perform connectivity tests (SSH, FASP, TCP or Aspera Orchestrator) for a particular remote node, on a selected port(s).

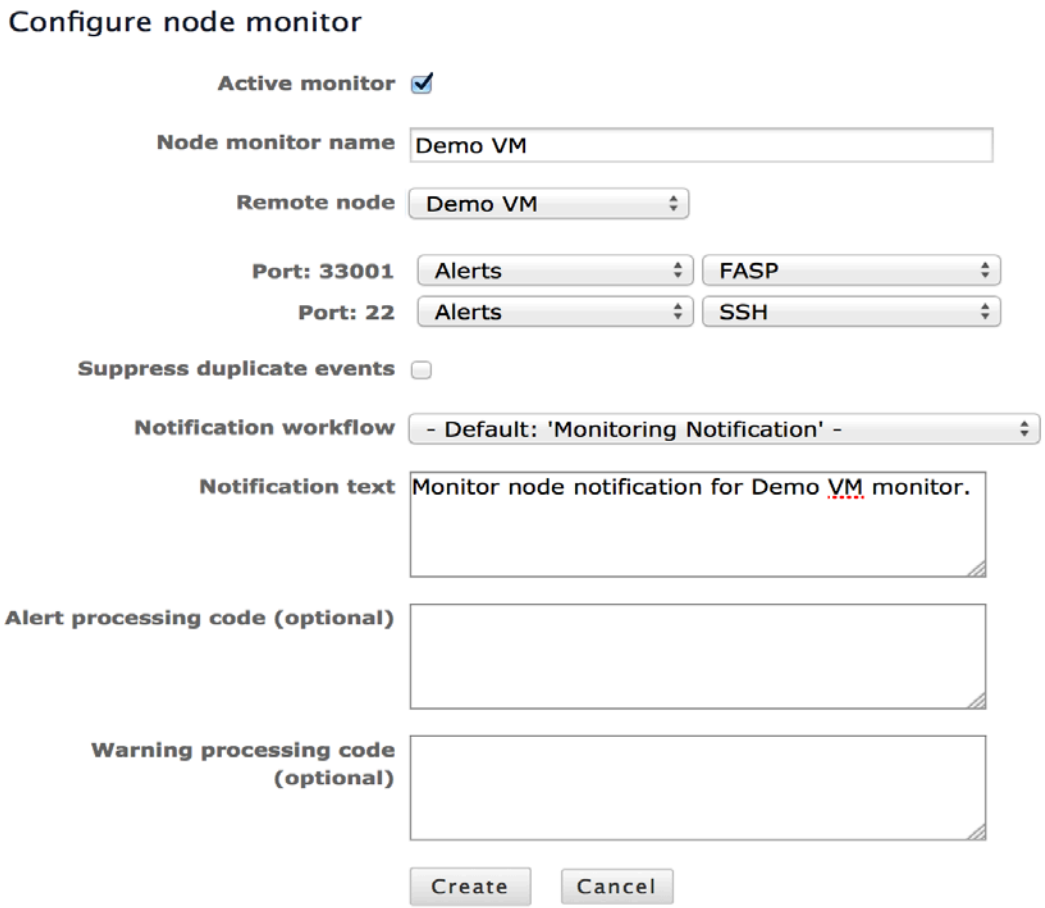
To add a node monitor, select the Monitor menu, and then click the "Configure nodes monitor" button. This will take you to the screen where you can add your new workflow monitor.



Click the "New nodes monitor" button.



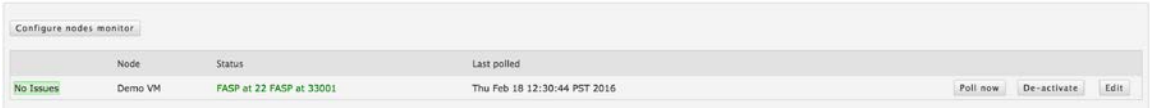
This takes you to the screen where you can define your new nodes monitor. In the following example, a node monitor is created for a remote node (configured under Workflows - > Remote Nodes)



Option	Description
Active monitor	If this check box is checked, the node monitor is created in active mode. Meaning the monitor of the node will be polling for the status of the nodes when the monitor is active.
Node monitor name	The name of the monitor node name.
Remote node	The remote node on which the monitor is created. The list comes from all the remote nodes configured on Orchestrator.
Port	This field displays all the ports that are configured while configuring the remote node. For each port user can set the Notification type and Test type. For example, in the above screenshot for port 33001 Alerts on FASP connectivity and for port 22 Alerts on SSH connectivity is selected. **NOTE**: If no ports are configured in the remote node configuration, by default SSH port 22 for SSH, TCP and Aspera Orchestrator nodes and default FASP 33001 for FASP node are associated to the monitor. Options for Test types are – SSH, FASP, TCP and Aspera Orchestrator. For Aspera Orchestrator, one orchestrator node can know the process statuses (i.e. statues of engine, mongrels, workers etc.) of another orchestrator.
Suppress duplicate events	If a monitor state doesn't change for a while, this flag stops a notification being generated at every check.

Notification workflow	This option lists all the workflows, present in Orchestrator. Select the one that will be triggered when the notification has to be sent. For example, an email notification workflow can be configured and assigned to the monitor. For all the notification, that workflow will be triggered with status of the monitor.
Notification text	The text content, indication some information on the monitor, For example Monitor name can be given in the notification text. This text will be sent in all the notifications that are sent concerned parties.
Alert processing code	This is an optional input. When there are alerts on a particular node for the selected test, alert processing code is executed mostly in lieu of amending the situation that caused the alert.
Warning processing code	This is an optional input. When there are warnings on a particular node for the selected test, warning-processing code is executed mostly in lieu of amending the situation that caused the warning.

When there are not issues, the notification pane shows 'No Issues'. This means that all the respective tests on all the ports are responding. For example in the below notification (for the example created above) SSH connectivity and FASP connectivity on those ports are listening and are active. When the connectivity test is successful for the particular port, the Status for the port appears in green. In the case when the connectivity fails, the label will turn red indicating that the connectivity for that port is failing.



In case there are Warnings, 'Warnings' label appears with yellow background. Similarly, in case there are Alerts, 'Alerts' label appears with red background.

The screen also shows few more buttons.

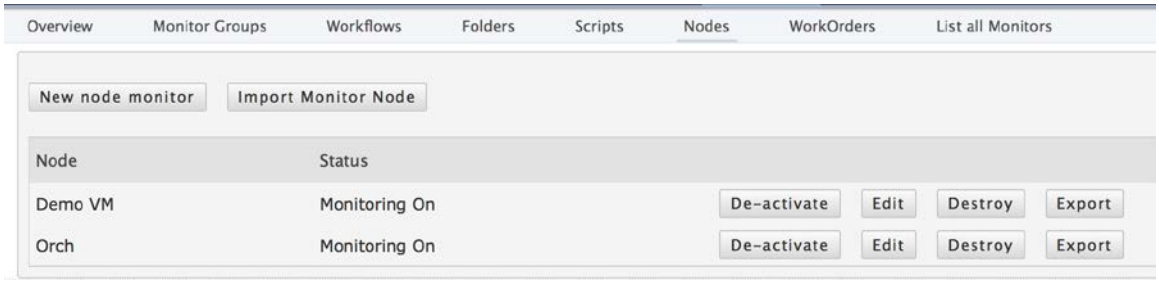
Poll now – can be used when force poll the monitor.

De-activate – will deactivate the monitor. The node monitor will no longer poll for connectivity tests and the status will become 'Not running.'

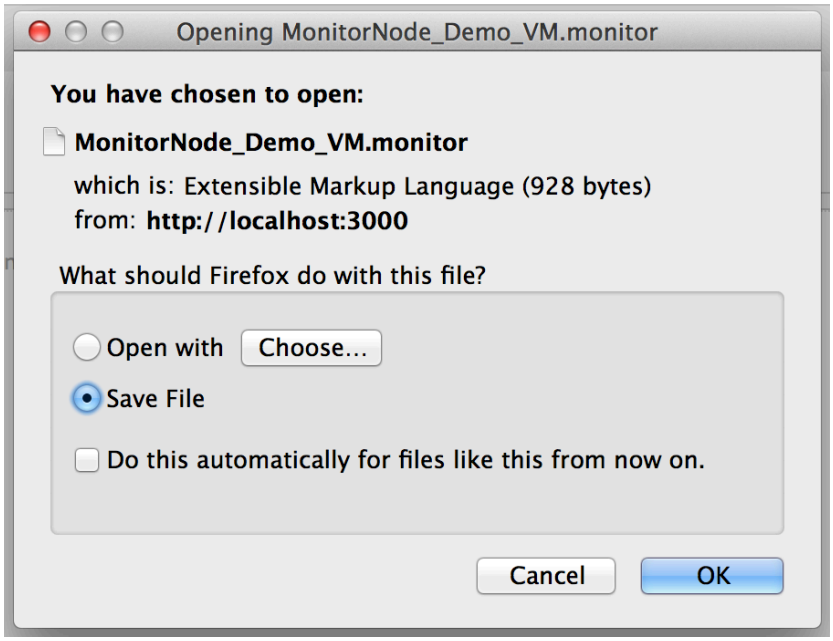
Edit – Edit button will enable user to edit the monitor configuration. For example the name of monitor, or change the notification type from 'Alerts' to 'Warnings' for a port or even change of the test type

NOTE: The remote node associated with the remote node cannot be changed once set. This would essentially means that if the user wants to change the remote node on the monitor, he/she will have to create a new node monitor and configure the desired remote node in the new node monitor.

HOW TO EXPORT A REMOTE NODE MONITOR



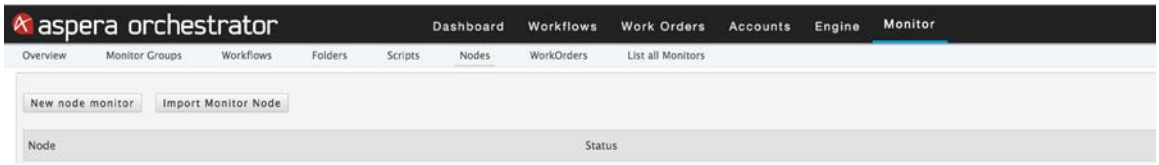
Navigate to Nodes tab and click on the 'Export' button.



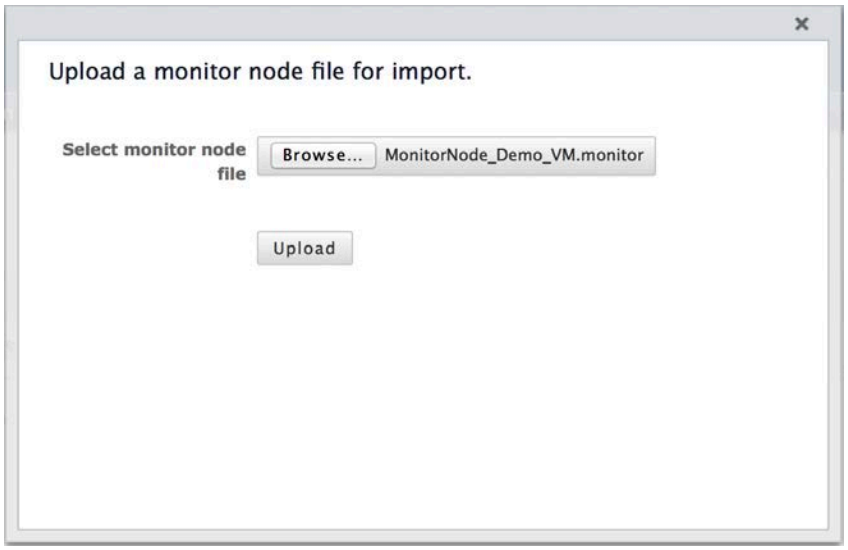
Click in 'OK' for the prompt. This will create and save a .monitor file with monitor name as the name of the file.

HOW TO IMPORT A REMOTE NODE MONITOR

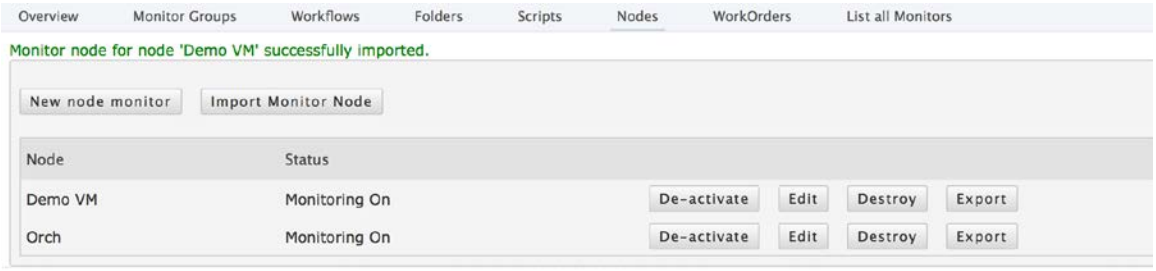
Navigate to Monitor menu and click on the Nodes tab. Click on the 'Import Monitor Node'.



Select the .monitor file and click on 'Upload'.



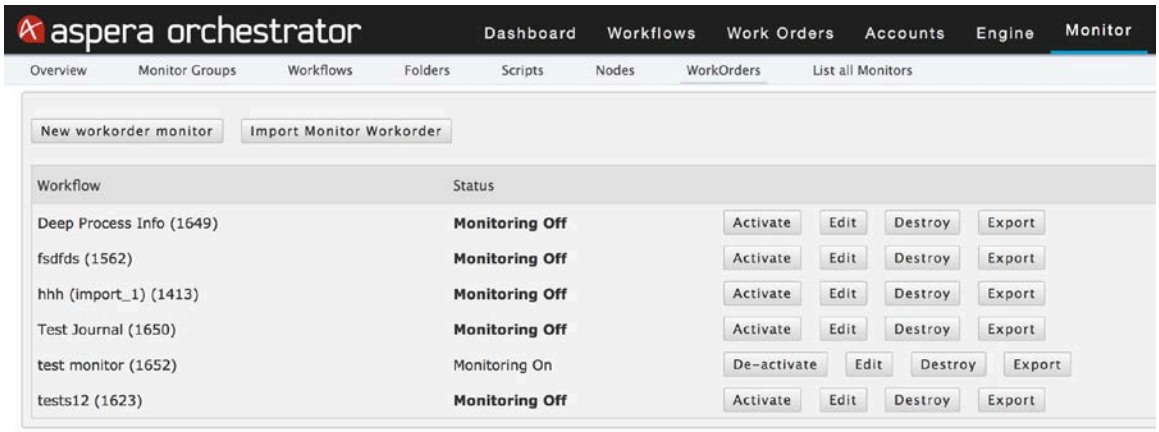
Upon successful importing of the monitor, below screen will be displayed.



Note: Only files with extension as '.monitor' are accepted as valid monitor files.

HOW TO ADD A WORKORDER MONITOR

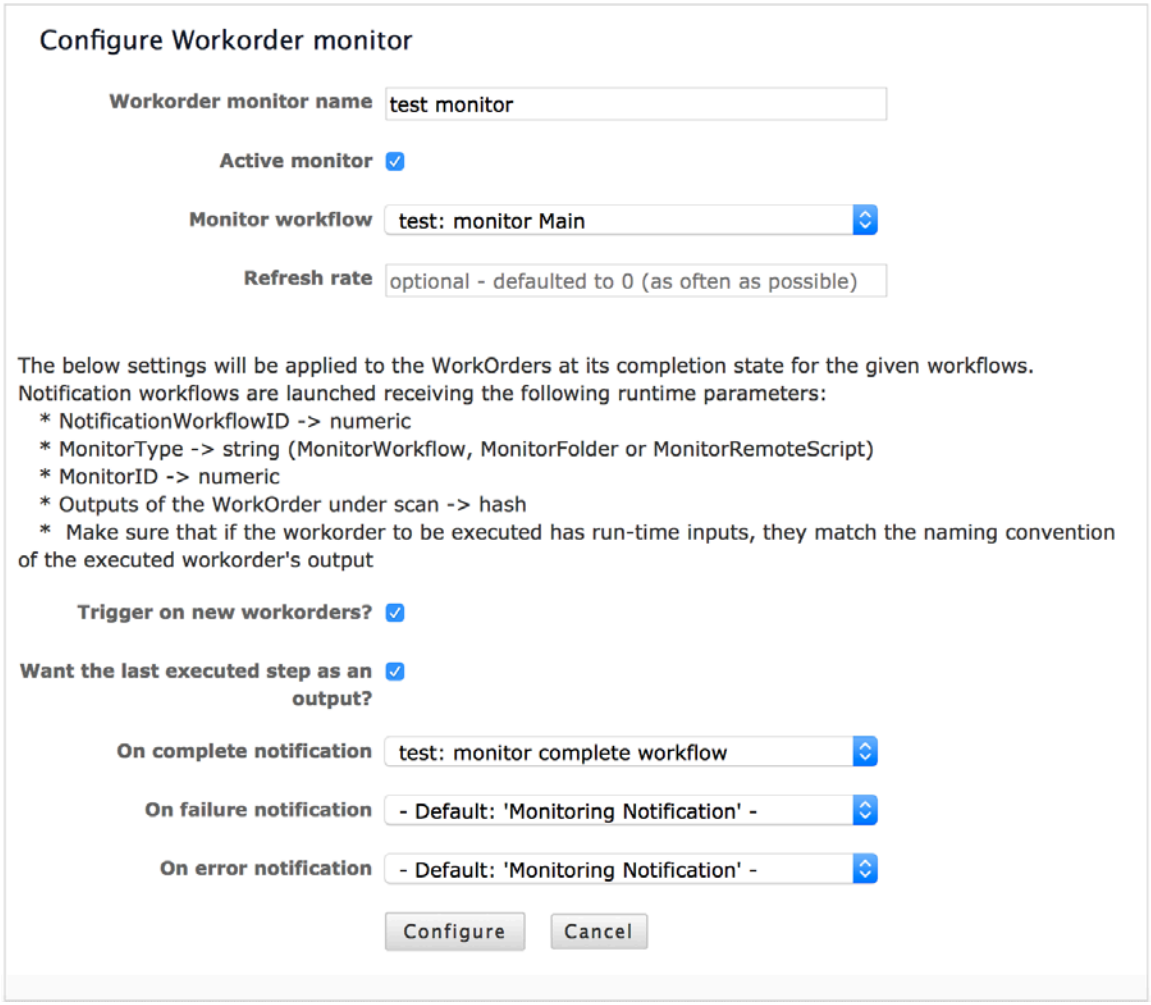
Select the Monitor menu, and then click the "Configure workorders monitor" button. This will take you to the screen where you can add your new work-order monitor.



Monitor Work-orders are used to monitor the workflow instances and to perform an action on its Completion / Failure / Error.

Options described

Option	Description
Active Monitor	The Monitor process will add the new monitor to its list of workflows to monitor
Trigger on new work-orders	If this is checked this monitor will only start triggering on new instances of work-orders created after the monitor is activated.
Want the last executed step as an output	This gives the monitoring work-orders output as a hash parameter called "last_executed_steps" storing the step_id and step_name as output.
Completion Notification, Failure Notification, Error Notification	If any workflow has to be triggered on action complete/ Failure/ error that can be specified here

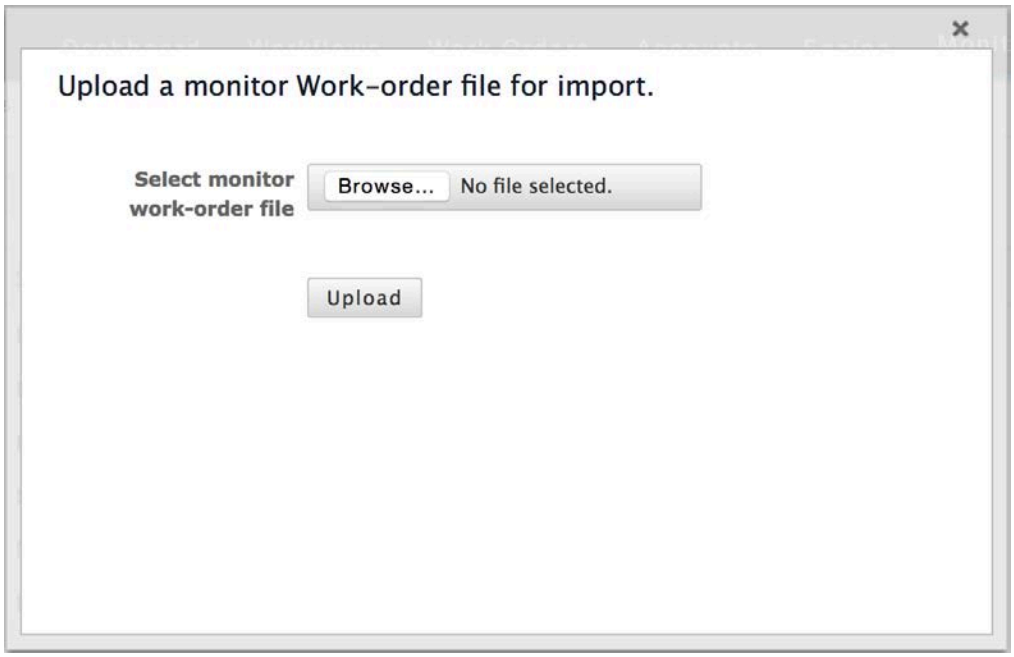


Import and export features are provided for all the monitors.

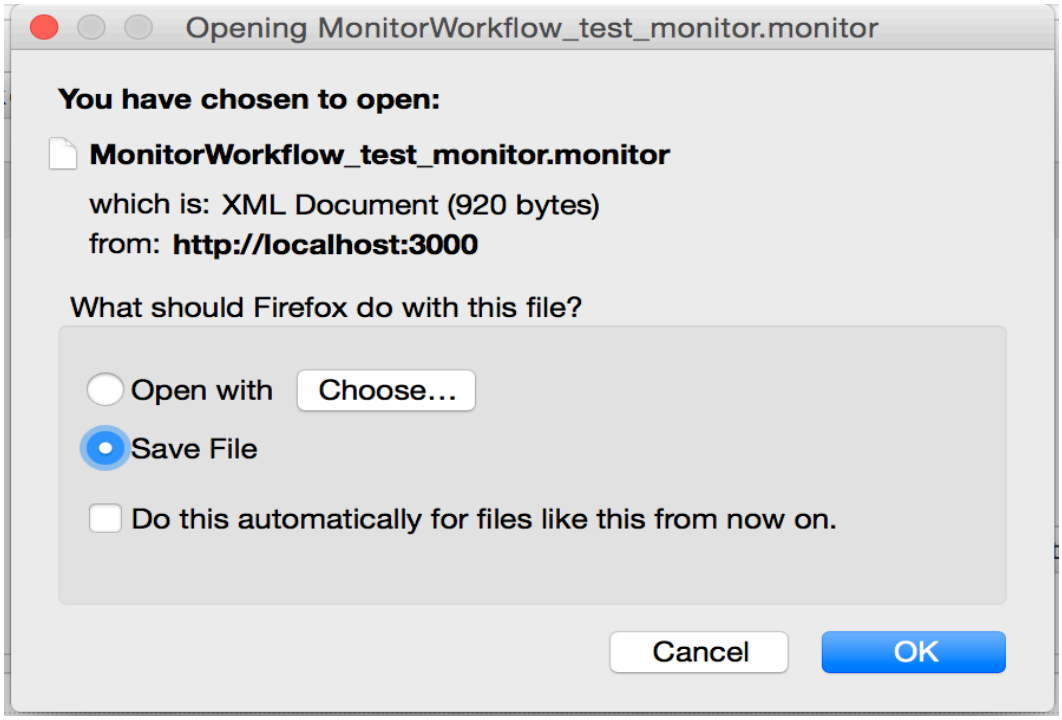
Note: Please specify the runtime parameters needed for the notification workflows in the below format since the mapping happens automatically in the code.

Step_name:field_name

Import screenshot for a monitor work-order is shown below.



Export screenshot for a monitor work-order is shown below.



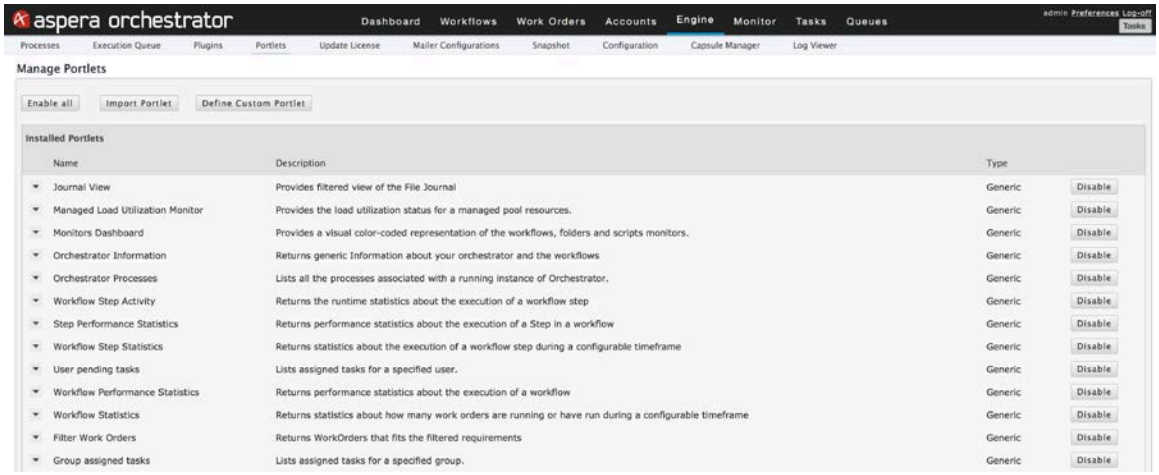
DASHBOARD AND PORTLETS

The Dashboard tab in Orchestrator User Interface is reserved for portal pages to be created using the existing portlets or a custom portlet. Portal pages can be used to

- Showcase data within Orchestrator collected by various work orders, workflows and processes
    - The data can be anything from statistics of a work order to a UI for resources or queues
- Interact with systems outside Orchestrator (e.g. Aspera Faspex WEB UI)
- Start work orders with run-time parameters being provided from UI

HOW TO ENABLE/INSTALL A PORTLET

Navigate to Engine -> Portlets

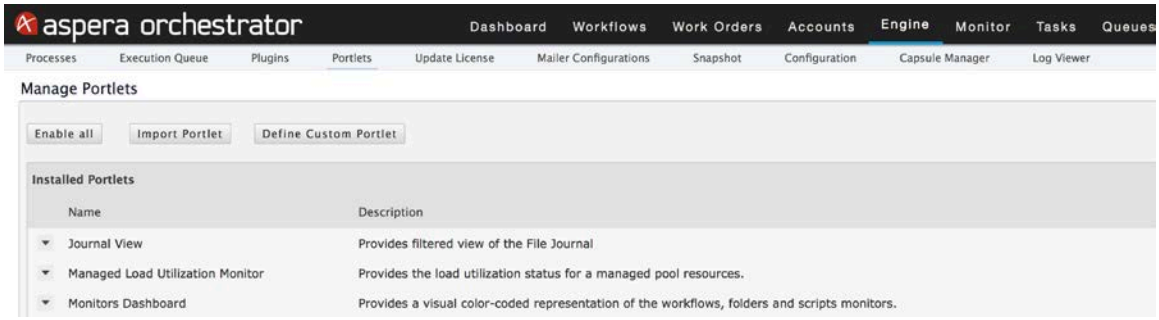


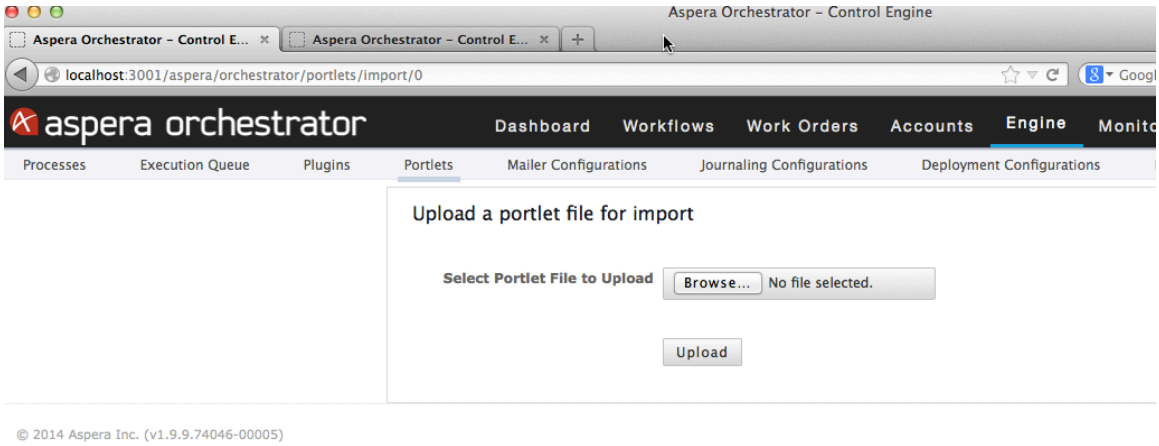
The page is divided into two sections – Installed Portlets and Other Portlets. If a portlet is already enabled it is ready to use, else Enable from the Other Portlets section.

HOW TO IMPORT A PORTLET

Navigate to Engine -> Portlets

Click Import button to obtain a page to import a portlet.





Click Browse to the path where the portlet file is downloaded. Once selected, click Upload.

Find the Portlet installed under the Engine -> Portlets

HOW TO MODIFY EXISTING PORTLETS

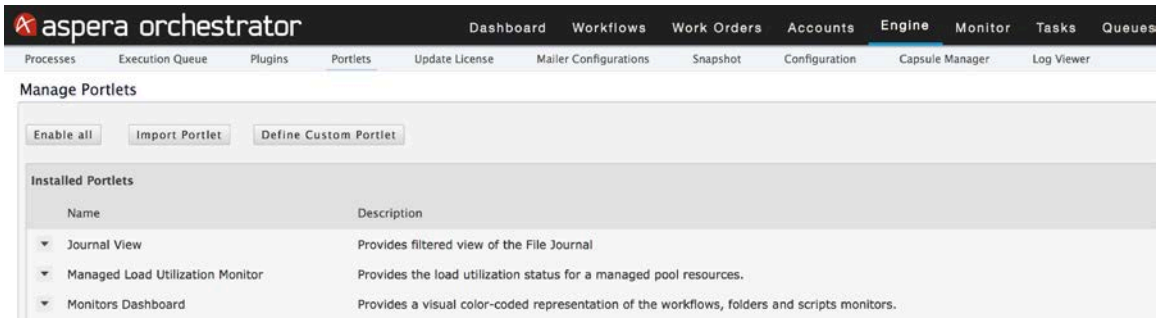
Note: This should be done by users with knowledge of Ruby and Rails and who have gone through the Orchestrator Portlet training.

Traverse to the portlet path C:\Program Files (x86)\Aspera\Orchestrator\portlets\ on Windows or /opt/aspera/orchestrator/portlets/ on Linux. Find the portlet that needs modification and traverse into that directory.

Select the <portlet_name>.rb file for Ruby changes and open in Notepad or other file editing tools. Select the <portlet_name>.erb file for Rails/HTML changes. Make necessary changes and save. After making the changes, Navigate to Engine -> Portlets in Orchestrator UI and disable + enable the portlet for the new changes to reflect.

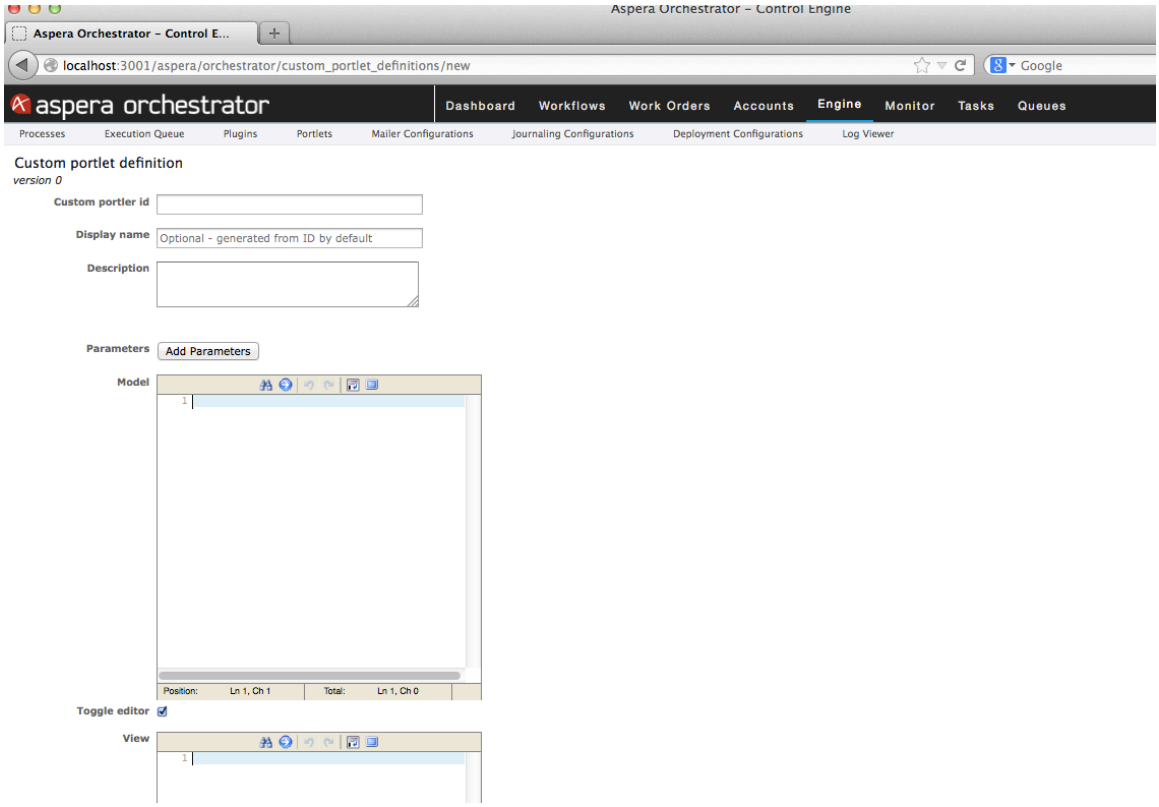
HOW TO CREATE A CUSTOM PORTLET

Navigate to Engine -> Portlets. To create a new custom portlet, click the 'Define Custom Portlet' button.

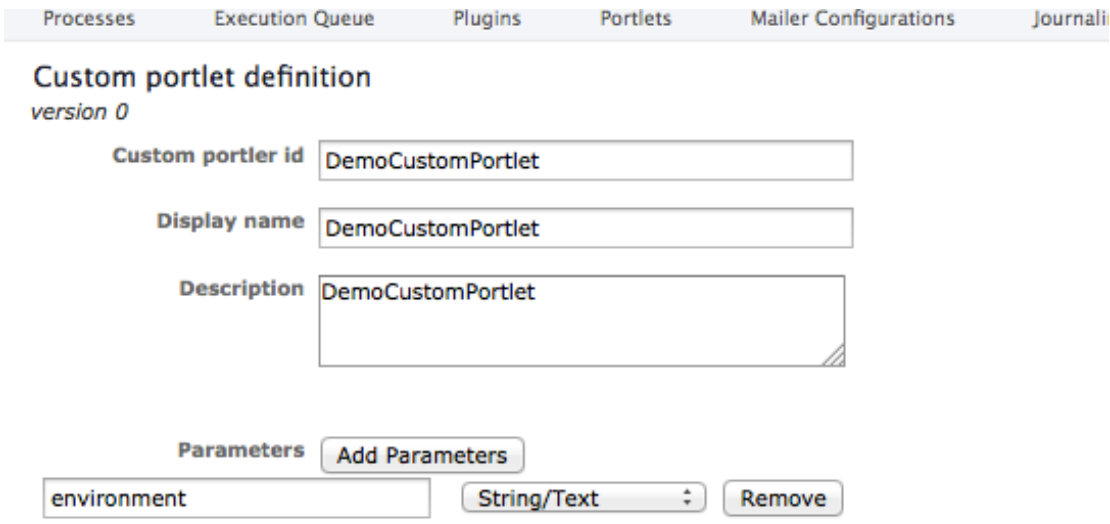


EXAMPLE OF A CUSTOM PORTLET

As an exercise let's build a custom portlet to display the Users in Orchestrator. Navigate to Engine -> Portlets. Click of Define Custom Portlet to see a web page with two text areas.



Provide an ID, display name and description name to identify the custom portlet. Add parameters using the "Add Parameters" widget, if any parameters are needed.



In the Model field enter the method or methods that your view (HTML page | Portal Page) needs to do its rendering.

Enter below code in Model text box

```
def get_orch_users
        begin
          users = User.find(:all , :select => "login")
        rescue Exception => e
          error "Could not fetch users in Orchestrator: #{e.inspect}"
          return []
        end
        return users
end
```

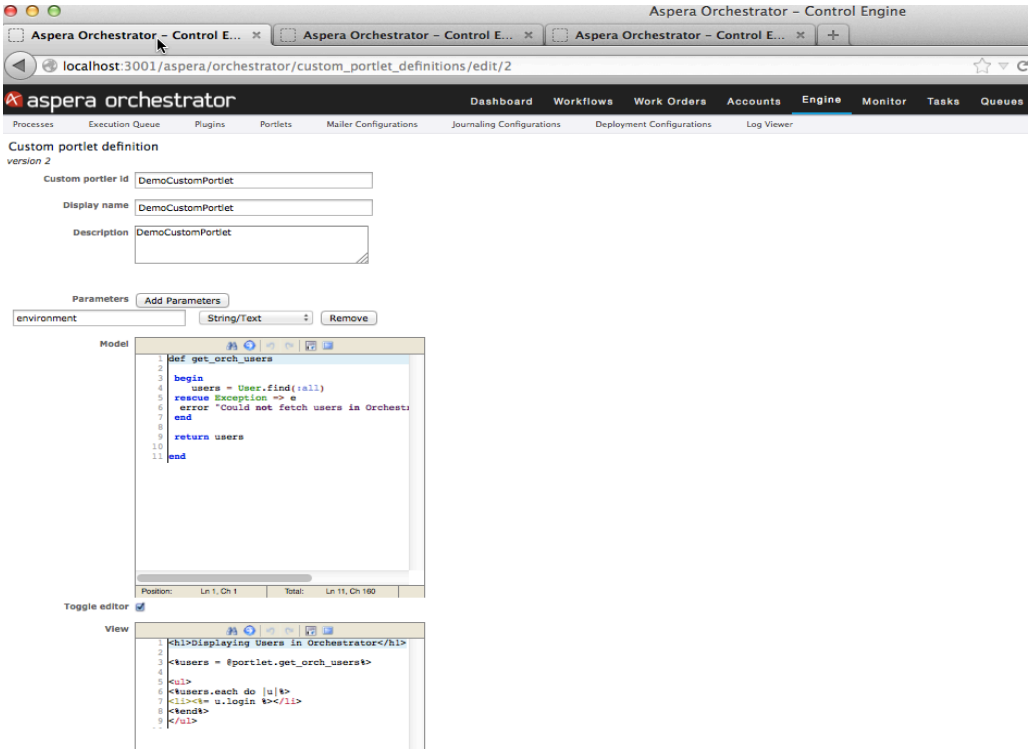
Enter below code in View text box

```
<h1>Displaying Users in Orchestrator</h1>
<%users = @portlet.get_orch_users%>
<ul>
<%users.each do |u|%>
<li><%= u.login %></li>
<%end%>
</ul>
```

Click Save at the bottom to save the custom portlet configuration. Now visit the **How to create the Portal page** section to create a portal page with this custom portlet.

HOW TO EDIT A CUSTOM PORTLET

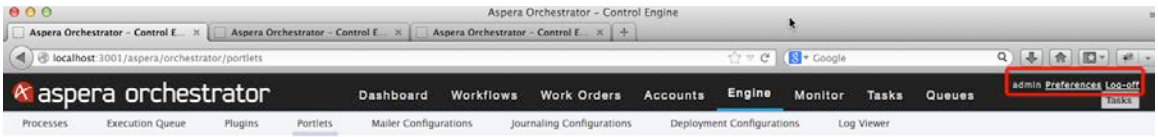
Navigate to Engine -> Portlets. Select the custom portlet that needs modification and click Edit item on the menu.



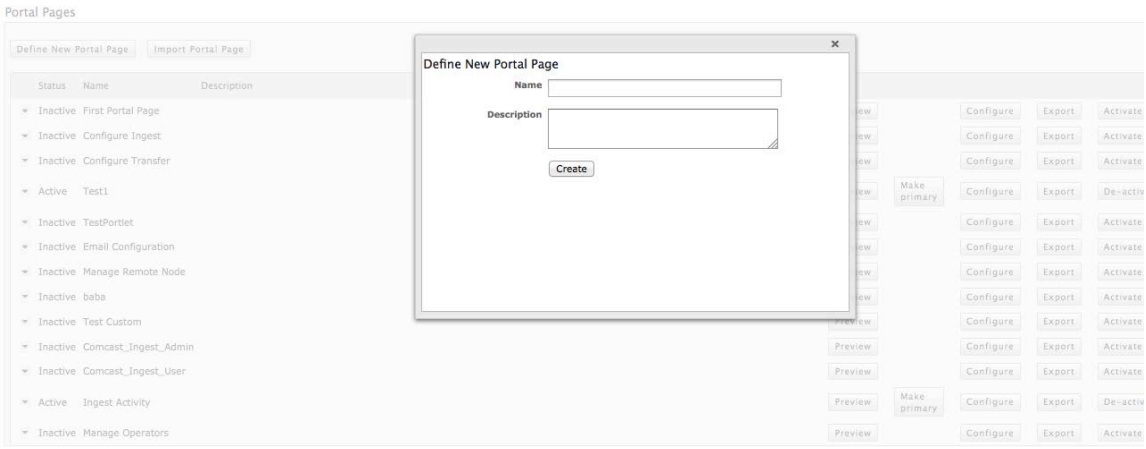
Click Save at the bottom to save the custom portlet configuration and refresh the custom portal page in Dashboards.

HOW TO CREATE A PORTAL PAGE THAT SHOWS UP IN DASHBOARD

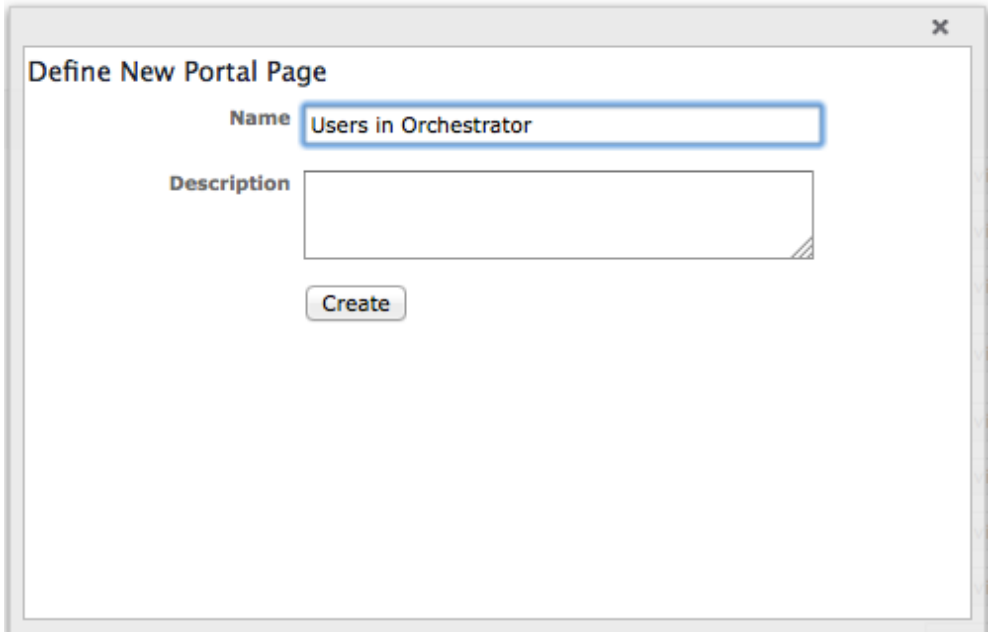
Click Preferences at extreme right of Orchestrator UI to visit User configuration.



The user information screen is rendered. Scroll to the bottom to find a Define New Portal Page button. Click it to see a popup asking portal page name and description.



Enter a name in the text box provided.



The portal page is now displayed in the table with other portal pages (if any). Click on Configure to select the content to display.

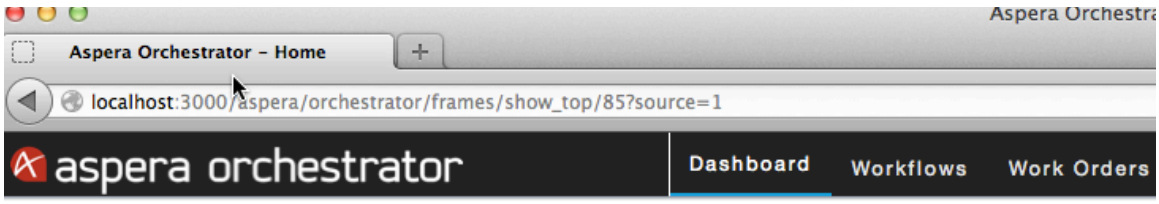
HOW TO CONFIGURE A PORTAL PAGE

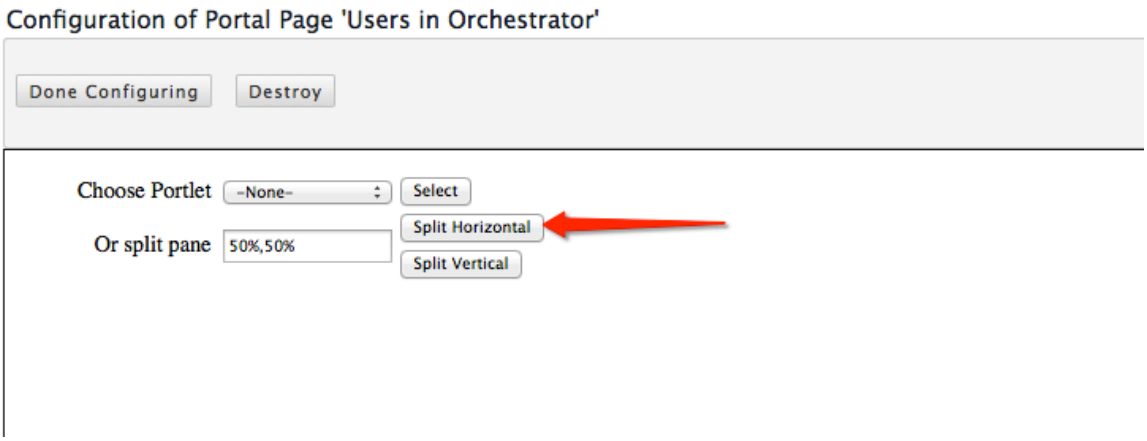
After creating a portal page, find it in the list of portal pages at the bottom of user preferences. Click Configure to select the portlet and customize the screen allocation.

Clicking on the buttons to the right can change position of a dashboard. '<<' will move the dashboard to the top and '>>' to the end. Up arrow button and down arrow button are used to move position to up or down by one.

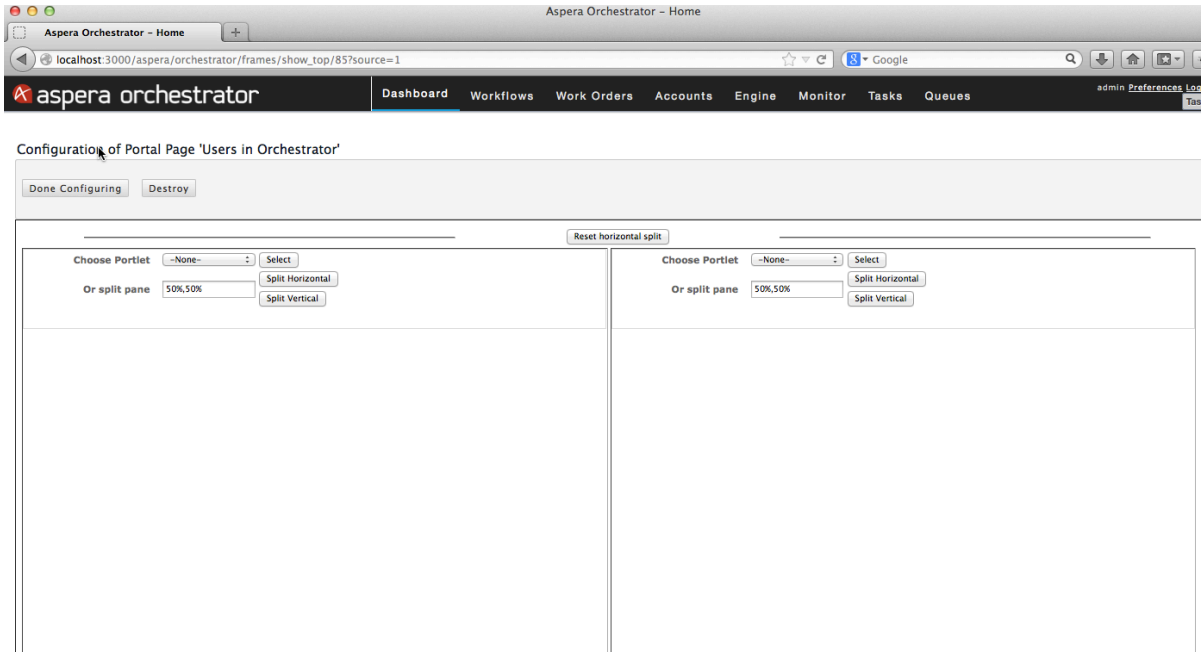
SPLIT THE PORTAL PAGE HORIZONTALLY

The configuration screen has provision to divide the page horizontally or vertically and a dropdown of available portlets. In the below example, the screen is split horizontally by clicking the Split Horizontal button with the page split exactly in half (50% each)



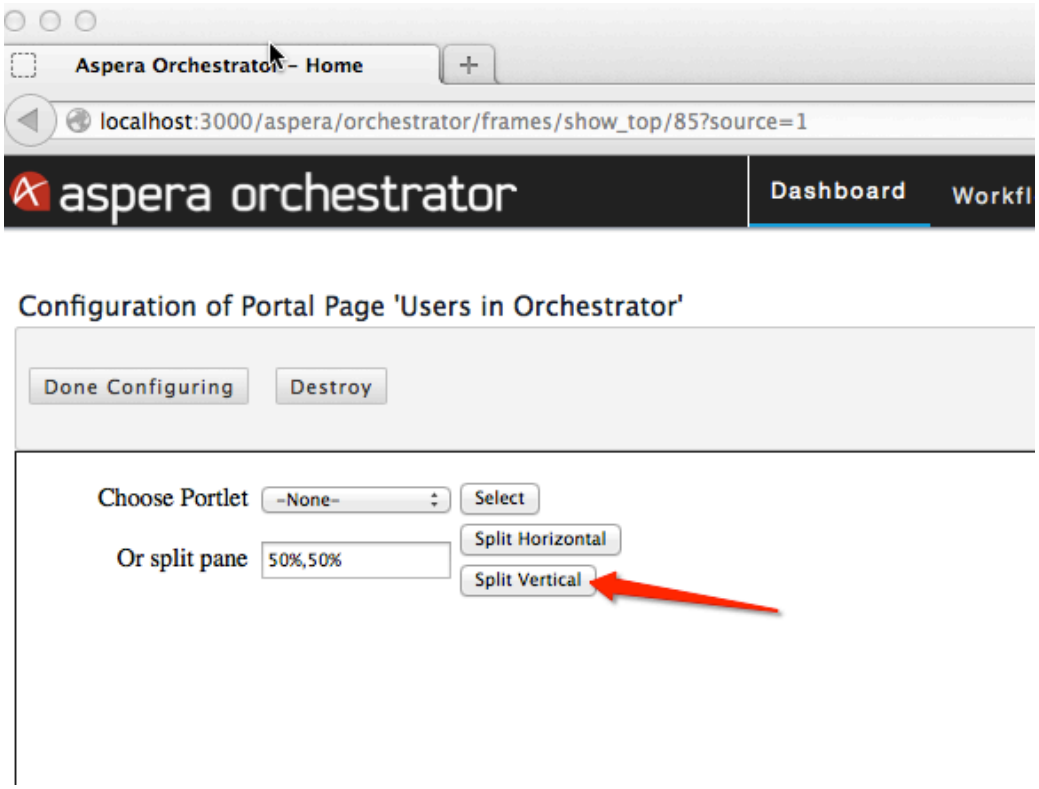


The resulting page will look like the below screenshot. The horizontal split can be reset using the Reset Horizontal Split button.

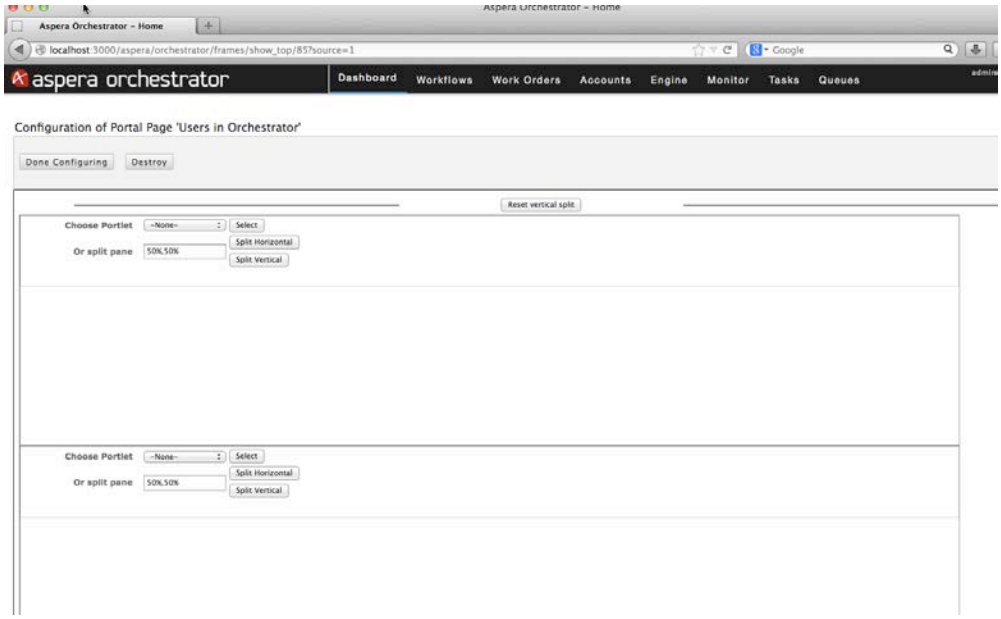


SPLIT THE PORTAL PAGE VERTICALLY

The configuration screen has provision to divide the page horizontally or vertically and a dropdown of available portlets. In the below example, the screen is split vertically by clicking the Split Vertically button with the page spilt exactly in half (50% each)

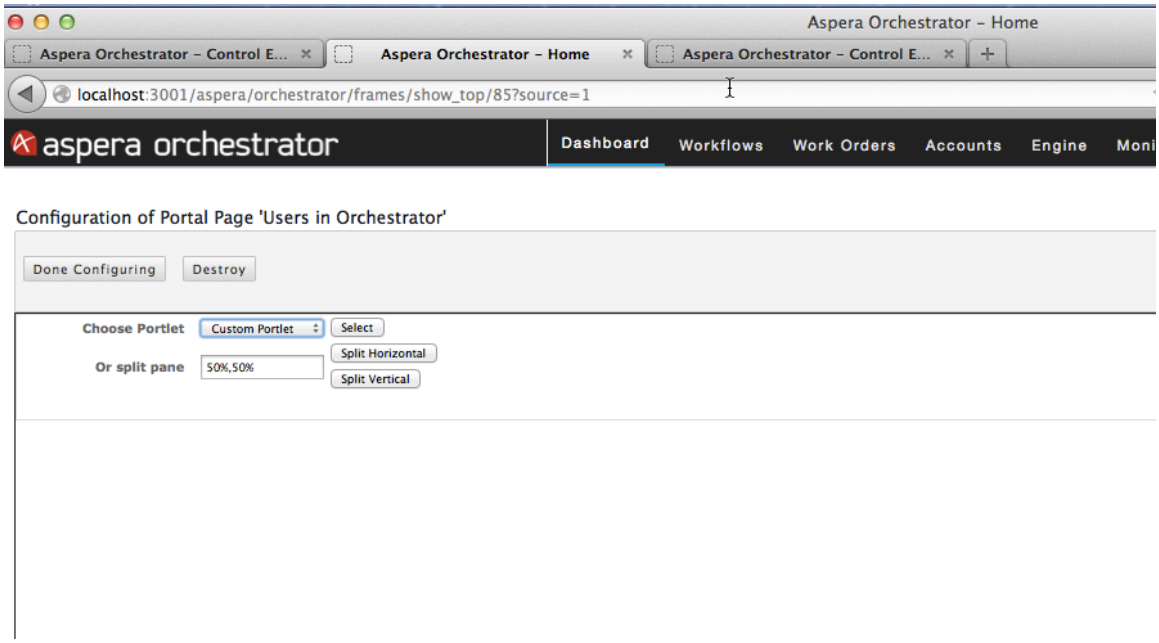


The resulting screen will look like below screenshot. Clicking the Reset Vertical split button can reset the vertical split.

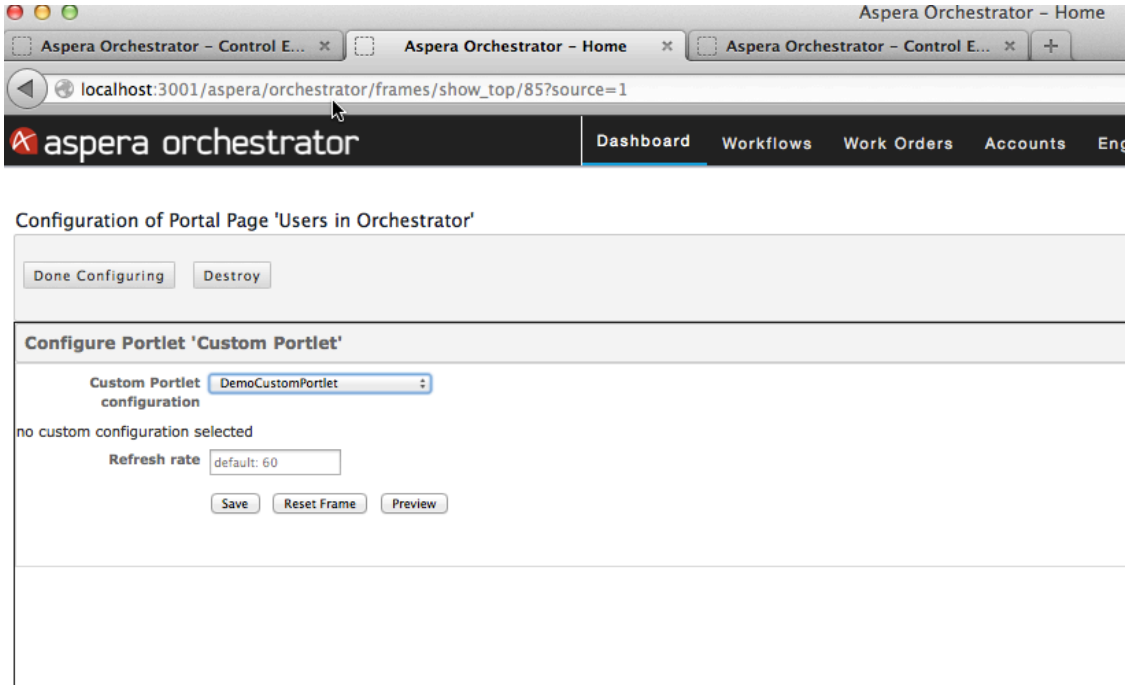


CHOOSING A CUSTOM PORTLET FOR PORTAL PAGE

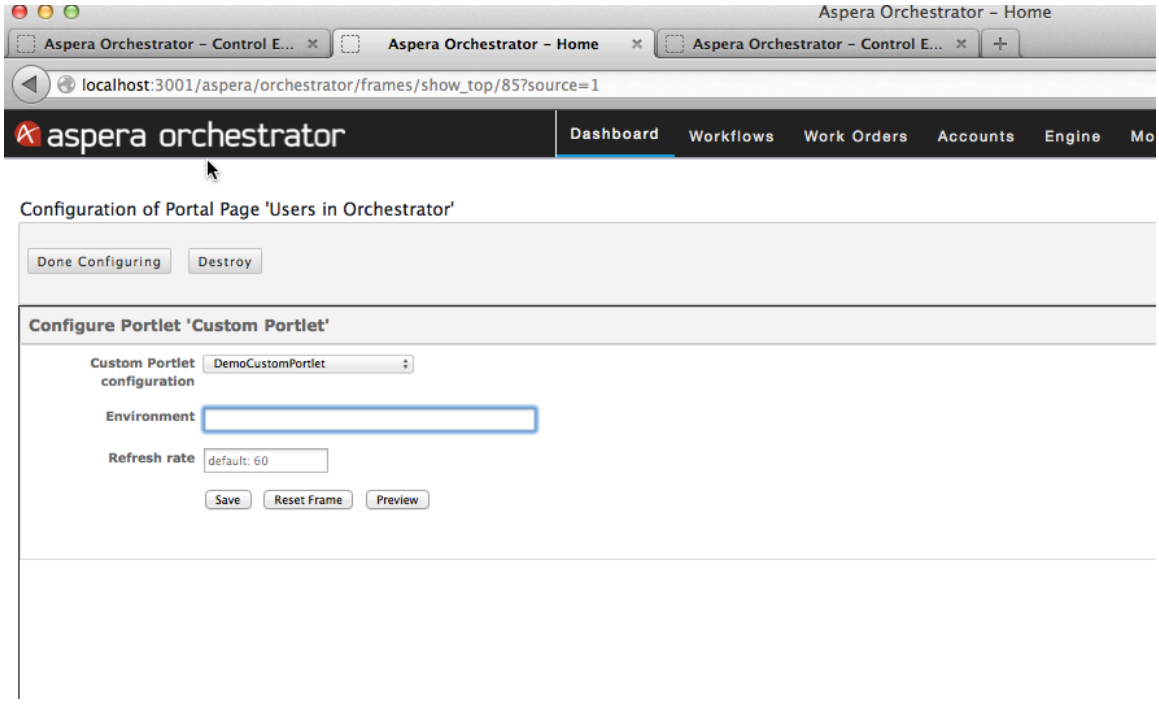
The procedure to choose a custom portlet is a bit different from choosing a generic/installed portlet. In the configuration screen of the portal page find the “Custom Portlet” in the Choose Portlet dropdown. Click Select



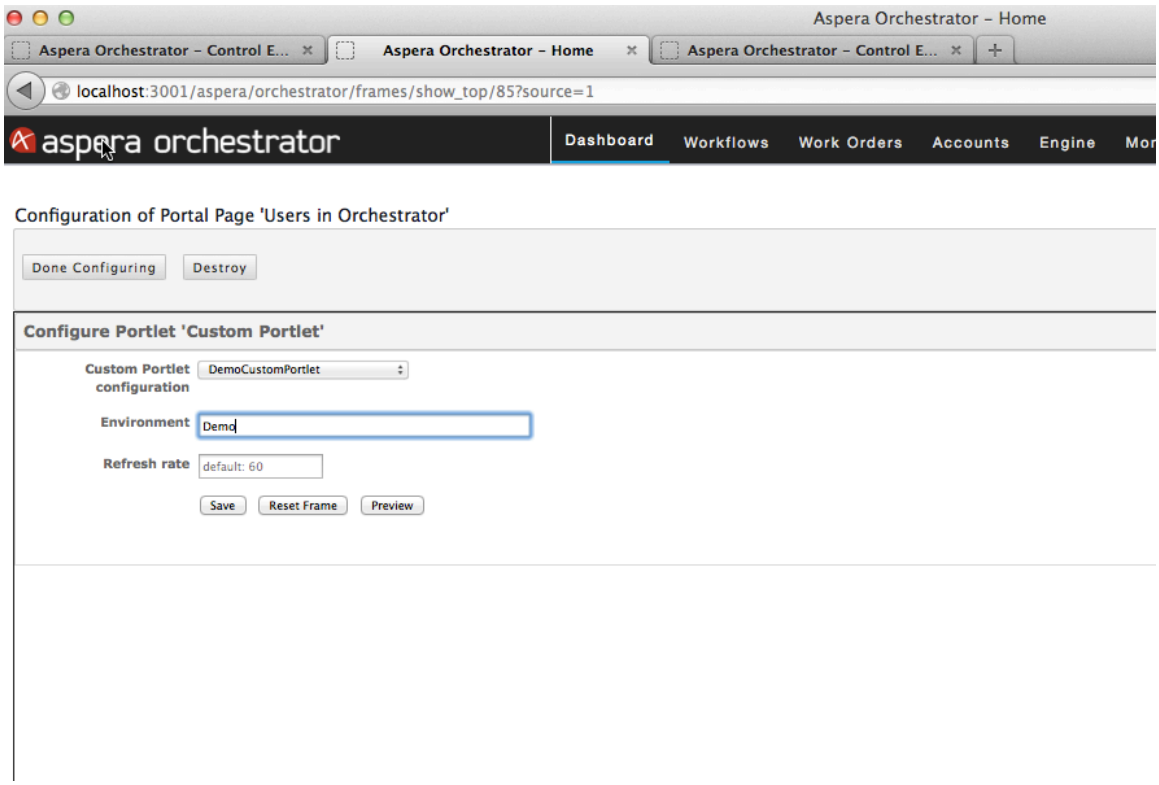
A dropdown of the Custom Portlets is shown, select the above developed DemoCustomPortlet



Click Save to obtain the runtime info regarding the CustomPortlet – in this case the runtime parameter Environment is displayed.

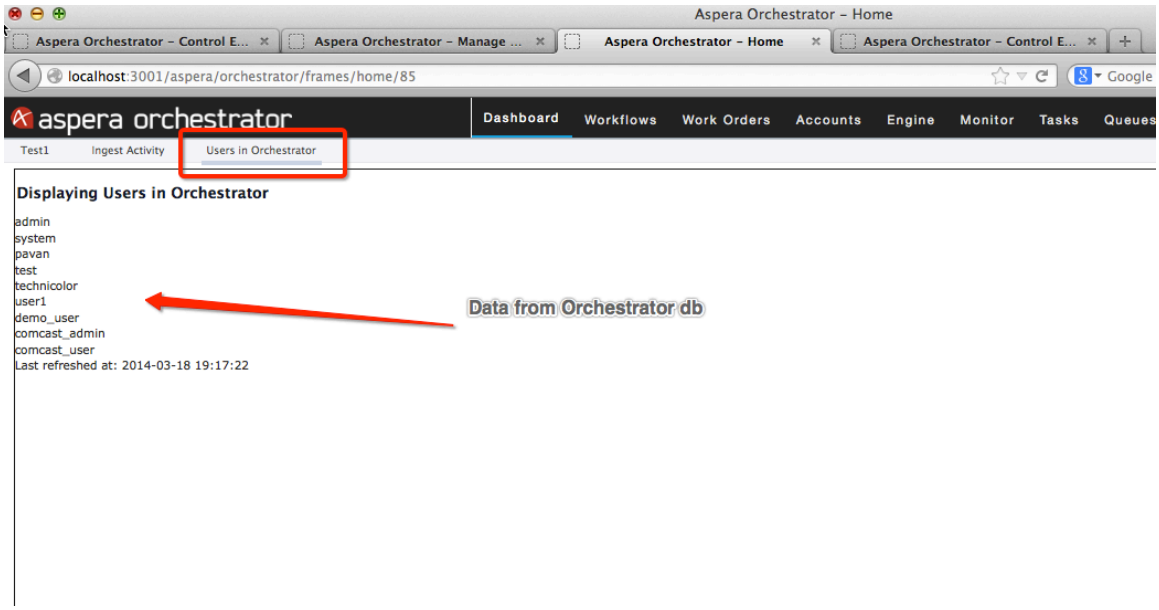


Enter a value for Environment and click Save button again. Your portal page using the Custom portlet is ready. Click Done Configuring button to go back to the user preferences page.



Click Activate link on the menu to make the portal page appear in the Dashboard. An option to preview the page also exists.

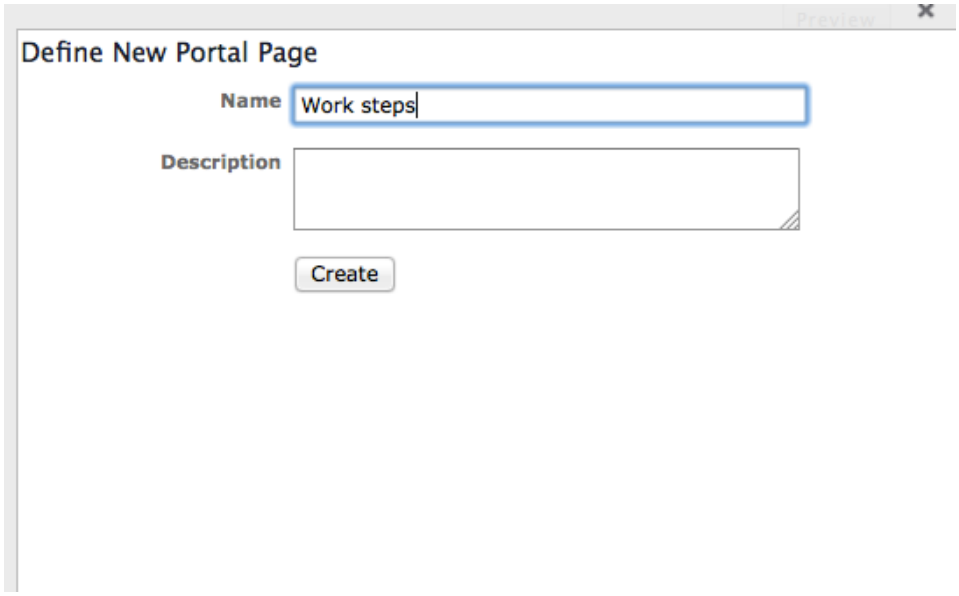
Navigate to Dashboard tab in Orchestrator UI. Find the portal page you just created in the secondary navigation.



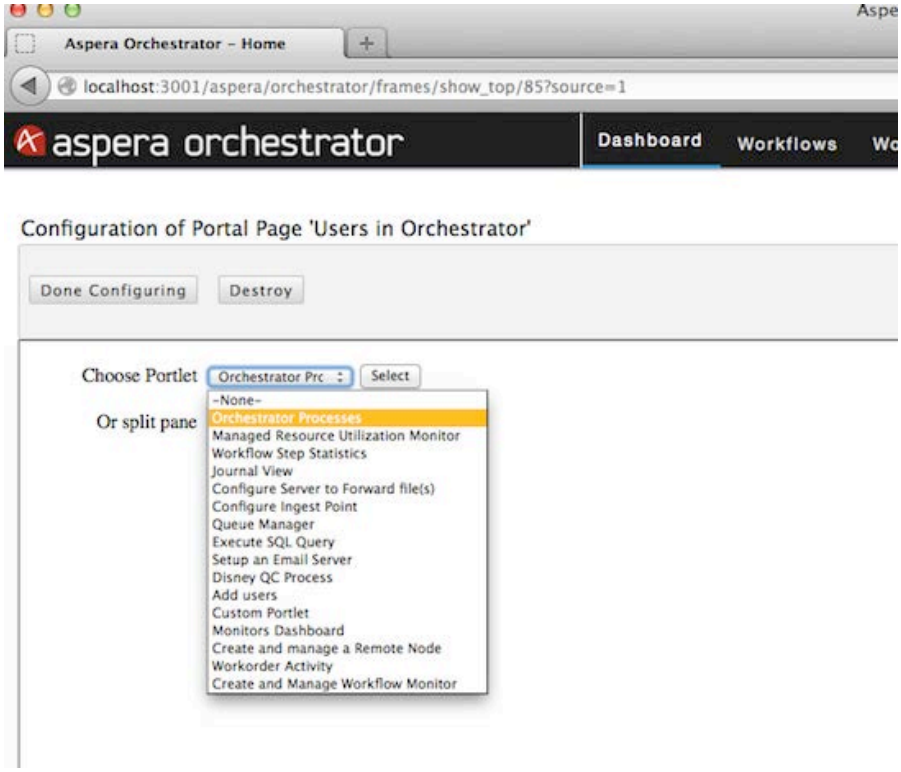
All the users currently in this test instance of Orchestrator are shown in the page.

CHOOSING AN INSTALLED PORTLET FOR PORTAL PAGE

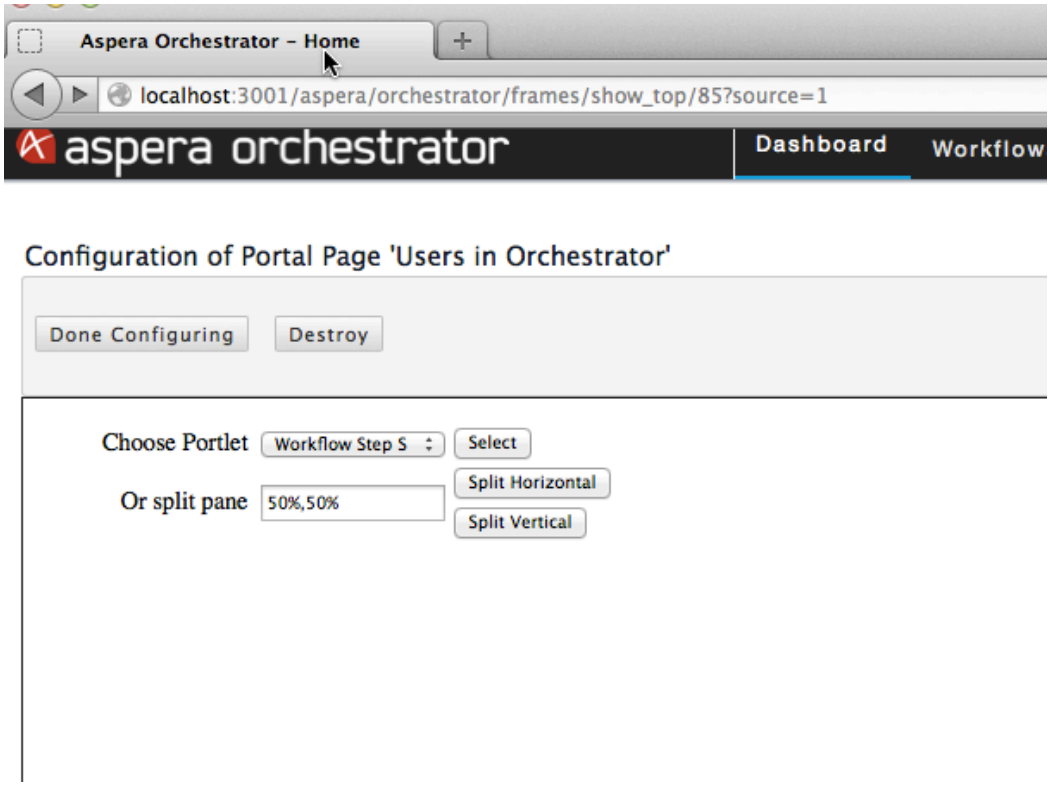
In the user preferences page, create a new portal page



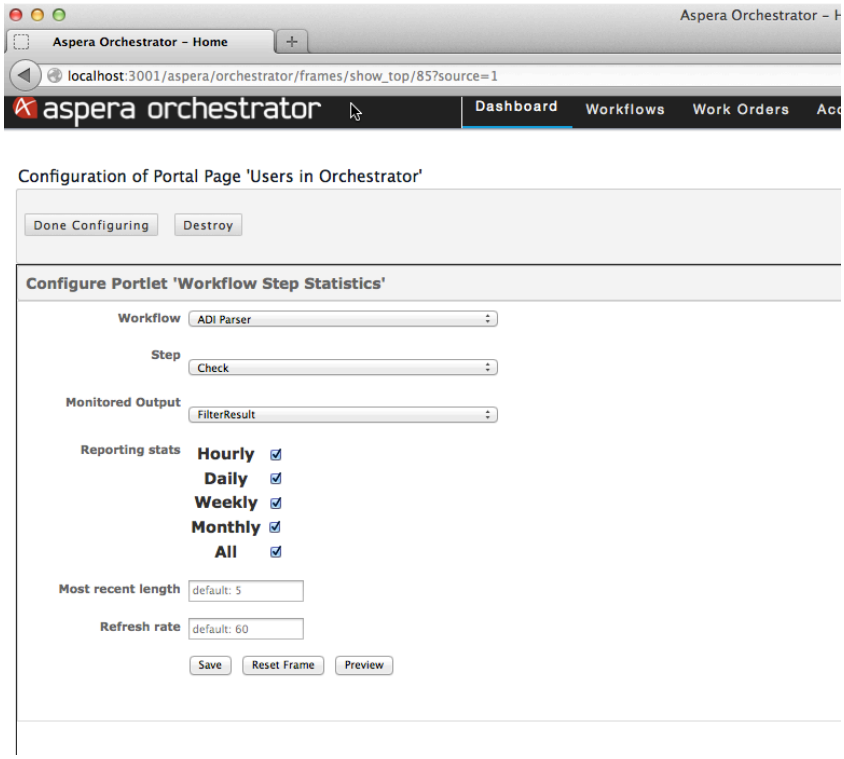
In the configuration screen of the portal page find the “Generic Portlet” in the Choose Portlet dropdown. Click Select.



As an example I have selected the Workflow steps portlet to be displayed in this portal page. Select it from the dropdown and click Select.



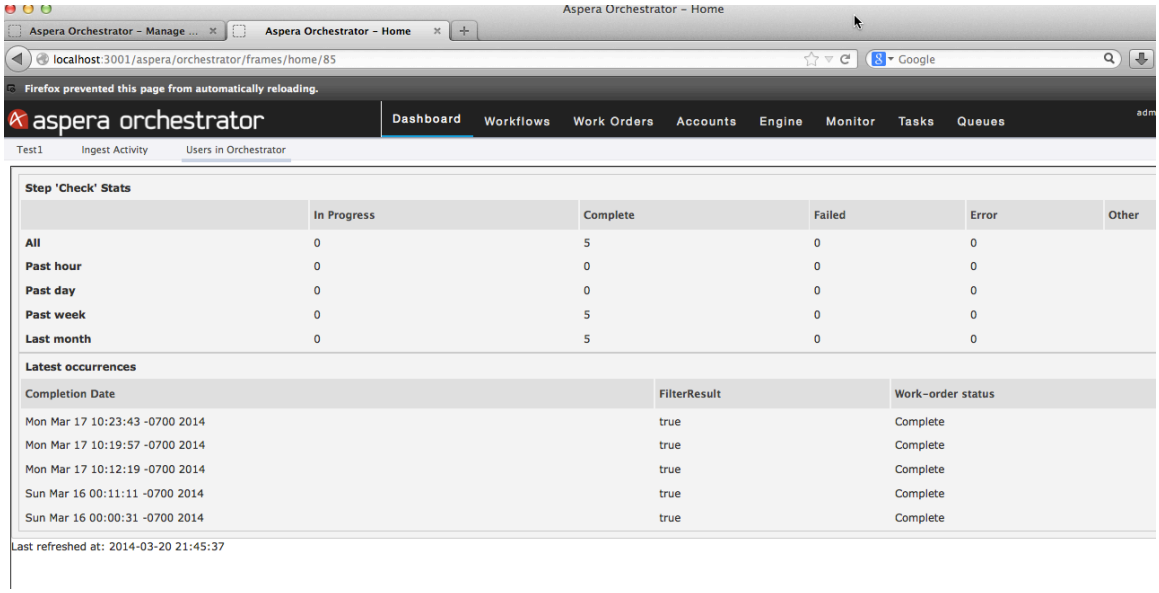
The workflow steps portlet has certain inputs that allow the portal page to be configurable. Select the necessary inputs.



Click Save button.

Click Done Configuring and Activate the portal page in User preferences page.

In the Dashboard tab find the portal page and click it to see it in action.

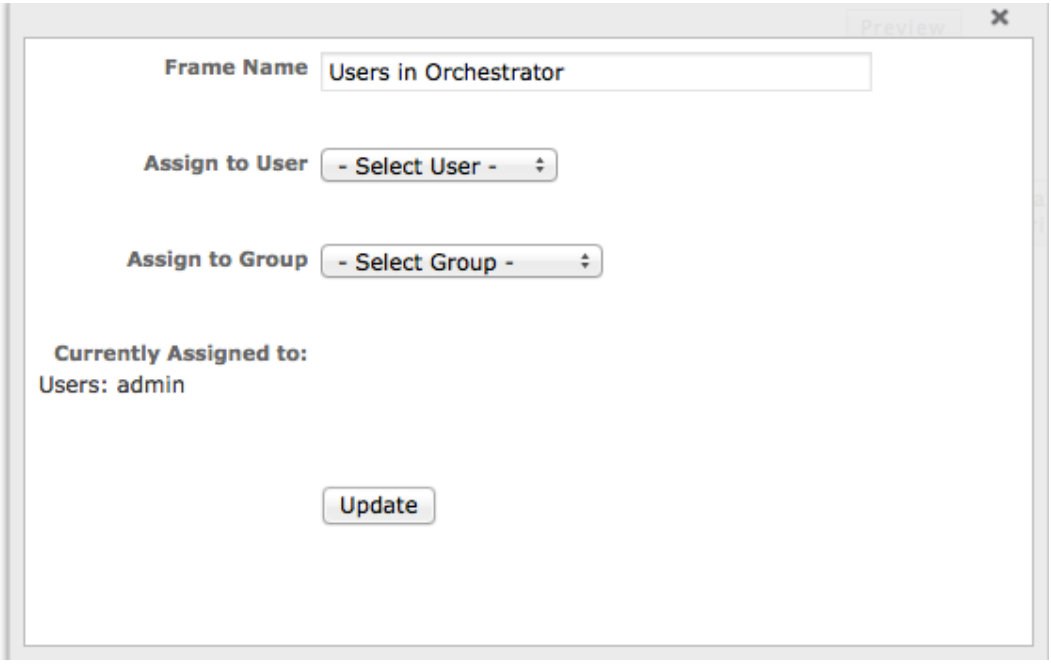


HOW TO ASSIGN A PORTAL PAGE TO OTHER USERS

A portal page needs to be created only once in Orchestrator by one user. It can then be assigned to other users who then need to Activate it.

On a completely configured and Activated portal page, click the down arrow to find a menu.

Click the Assign to a User or group on the menu (above).



Choose the User or Group or both and click update.

SYSTEM ADMINISTRATION

Differences between a Synchronous and Asynchronous worker

A Synchronous worker waits until the end of the execution of the step to gather its status and outputs. If timeout is left default and no unique message is rendered periodically there is a chance that the synchronous step fails after 60 seconds due to timeout error.

An Asynchronous worker gathers status and outputs of a step periodically. The frequency is determined by the polling frequency of the step (usually 5 seconds). There are less chances of the step timing out.

ORCHESTRATOR CONFIGURATION FILE

Orchestrator's configuration file is called orchestrator.yml. It contains the below keys which can be modified for enhanced configuration.

Key name	Default value	Description
active_active	false	Set value to true when Orchestrator is clustered as active-active.
ad_user_groups	Contributor	When AD user logs in, this group is assigned by default
allow_blank_password	false	Users with no password set will be forced to create a password
allowed_logon_attempts	3	User accounts will be deactivated after these number of attempts when users login with incorrect password
archive_dir	OS Specific (refer to Orchestrator install directories section)	This directory contains an archive of plugins, capsules and snapshots, a result of actions initiated from Orchestrator UI
aspera_dir	LINUX: /opt/aspera Windows: C:\Program Files (x86)\Aspera\Orchestrator\www\	The parent folder for all Aspera products.
asynch_threads	2	Determines the number of Asynchronous workers that are spawned when Orchestrator starts.
auto_activate_portlets	false	When a portlet is assigned to another user or group they will be activated, if this is set to true. Default behavior is users needed to activate the portlets manually.
build	N/A	Represents the Orchestrator build number installed. Should not be edited,
cleanup_cutoff	30	Determines the number of days the successful work orders' data is kept in the database.

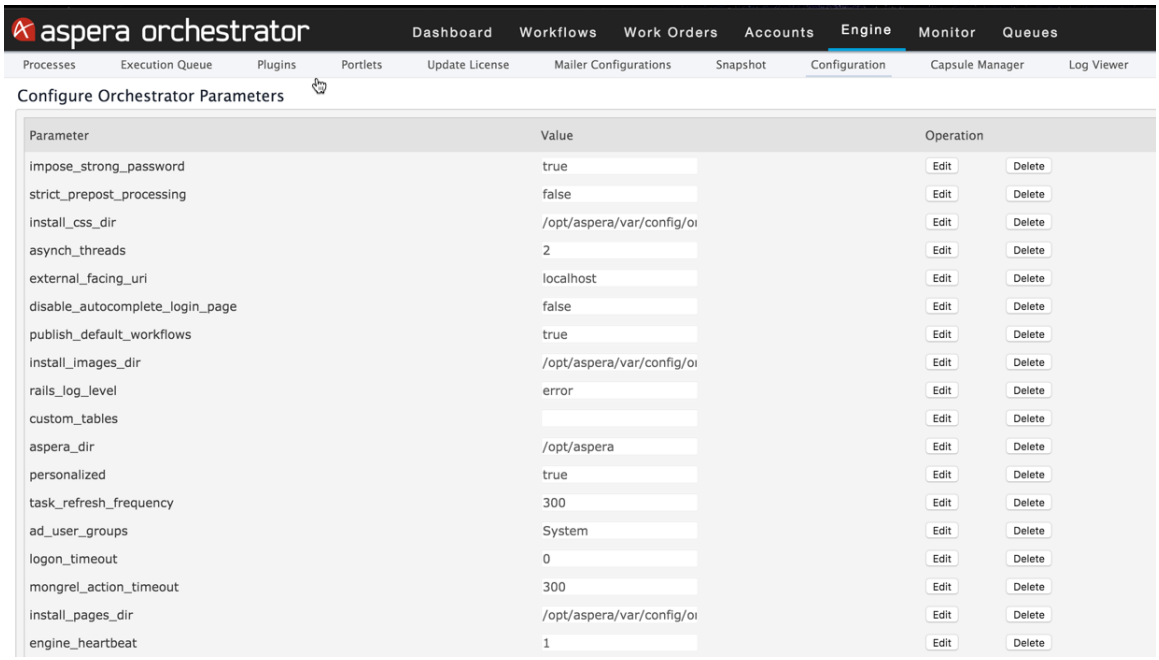
config_dir	OS Specific (refer to Orchestrator install directories section)	This directory contains actions folder which has runtime instances of the plugins, orchestrator.yml config file, database.yml database configuration file, licenses folder, capsules folder, actions_config folder, default_plugins_list.yml file, default_portlet_list.yml file
custom_tables		Any custom database tables to be backed up should be entered here as comma separated values
date_format	american	Date format for dates displayed in Orchestrator
db_bin	LINUX: /opt/aspera/common/mysql/bin Windows: C:\Program Files (x86)\Common Files\Aspera\Common\mysql_x64\bin	This parameter gives Orchestrator the path to the bin directory of MySQL for commands like 'mysqldump'
default_step_timeout	60	The number of seconds after which a step should fail after a period of inactivity
disable_autocomplete_login_page	false	By default the login page allows autocomplete of user name and password
engine_heartbeat	1	The frequency with which Orchestrator engine is polled for status.
engine_instance	0	This parameter takes a unique value in each of the nodes in an active-active or active-passive cluster. Leave default for single node install.
external_facing_uri	N/A	Hostname or IP address
external_javascript_lib		Empty by default. Any new JS libraries used must be mentioned here
fast_start	false	Changing the value to true on Windows prevents Orchestrator to start all processes at once, a move that saves computer resources from being utilized fully.
hide_preferences_for_groups		If a group name is provided here, the Preferences link on the top right of every page is hidden
ifv_skip_start_event	false	If set to true, inline file validation's start events are skipped
impose_strong_password	false	If set to true, will impose strong password policies like, checking strength of password while user creation and user change password operation, keeping track of unsuccessful login attempts and not allowing the user to set any of the previous three passwords as the new password.
install_config_file	Config Dir + orchestrator.yml	The path where this config file (orchestrator.yml) is stored. Usually a folder that is not overwritten during upgrades.
install_css_dir	Config Dir + /stylesheets/	A safe place to store custom CSS files that are protected even during upgrades.
install_images_dir	Config Dir + /images/	A safe place to store custom images that are protected even during upgrades.

install_pages_dir	Config Dir + /pages/	A safe place to store custom web pages that are protected even during upgrades.
journal_kept_duration	10	The number of days entries are kept in the Journal view.
journal_on	true	Changing to false deactivates the Journal configuration on an Orchestrator server.
load_basic_workflow_dir	Orchestrator Dir + /vendor/workflows	The directory that has basic workflows to load when Orchestrator starts.
load_default_workflows	true	Changing to false prevents Orchestrator server from loading the default workflows.
log_file	Orchestrator Run Dir + log/orchestrator.log	Path to Orchestrator log file.
log_level	debug	'debug' is the highest log level available. Other log levels, in decreasing order, are info, warn, error, fatal and unknown.
log_retention_days	7	Number of days after which orchestrator log files are deleted
logon_timeout	0	The value, in seconds, denotes number of seconds after which a user is logged off after inactivity. Setting it to zero will allow users to be logged until they manually logoff.
maintenance_interval	600	The number of engine heartbeats after which an Orchestrator maintenance task is spawned.
maintenance_heartbeat_interval	300	The frequency with which Orchestrator maintenance tasks are spawned.
make_view_by_workflow_default	false	If set to true, the Work Order view is changed to 'View by Workflow'
max_tasks_displayed	12	The number of tasks that should be displayed in the Tasks widget.
mongrel_action_timeout	300	Number of seconds after which Mongrel process is determined to be stuck on an action
mongrel_max_daemons	3	The number of Mongrel instances that should be spawned when Orchestrator starts.
mongrel_max_serve	0	Deprecated
mongrel_starting_port	3000	The starting port where the first Mongrel instance is spawned. Other instances will be assigned a port that is incremented by 1 per spawned instance.
orchestrator_dir	LINUX: /opt/aspera/orchestrator Windows: C:\Program Files (x86)\Aspera\Orchestrator	Orchestrator installation directory
perform_ad_operation	false	If set to true Orchestrator will perform non-reversible AD operations such as adding, modifying or deleting Active Directory users.
persistence_mode	disk	Persistence files are stored either on 'disk' or in 'db'
personalized	true	If set to false, the CSS is reverted to the v1 stylesheet (blue header)

publish_default_workflows	true	Changing to false prevents Orchestrator server from publishing the default workflows.
purge_cutoff	60	Determines the number of days the failed, cancelled and error work orders' data is kept in the database.
rails_log_level	error	Can be increased to warn, info and debug (increasing order). Can be decreased to fatal.
referer_hosts		To block a request from a bogus referrers put the hostname or IP address of Orchestrator server here
run_dir	OS Specific (refer to Orchestrator install directories section)	This directory contains runtime information and is populated with files/folders from plugins, work orders, portlets and also contains the log file.
strict_cache_control	false	If set to true, cache control header are placed on all web pages
strict_prepost_processing	false	When true, a failure in pre or post processing of a step will result in step's failure.
strong_password_regex	^(?=.{8,})(?=.*[a-z])(?=.*[0-9])(?=.*[A-Z])(?=.*[@#$%^&+=]).*$	The regular expression against which the password is checked while creating new user and while changing current user password. This will affect only when impose_strong_password parameter is set to true. The default RegEx expects that the password: 1. Should have minimum 8 character of length. 2. Should have at least one uppercase character 3. Should have at least one lowercase character 4. Should have at least one of these (@#$%^&+=) special characters.
synch_threads	4	Determines the number of Synchronous workers that are spawned when Orchestrator starts.
task_refresh_frequency	30	The frequency in which Orchestrator polls for a new task.
use_db_time	false	By default, system time is used for date format
use_new_designer	true	If set to false, Orchestrator designer screen will revert to use the plain background instead of the grid display.
web_root	/aspera/orchestrator/	Change this parameters to any meaningful URL and Orchestrator will request Apache to render Orchestrator UI at that URL.

CHANGE ORCHESTRATOR CONFIGURATION

Editing the orchestrator.yml file in the var/config folder of the Orchestrator installation can change orchestrator's global parameters. There is also a screen under the Engine -> Configuration tab that allows Admin users to edit the global parameters.



SNAPSHOT MANAGEMENT

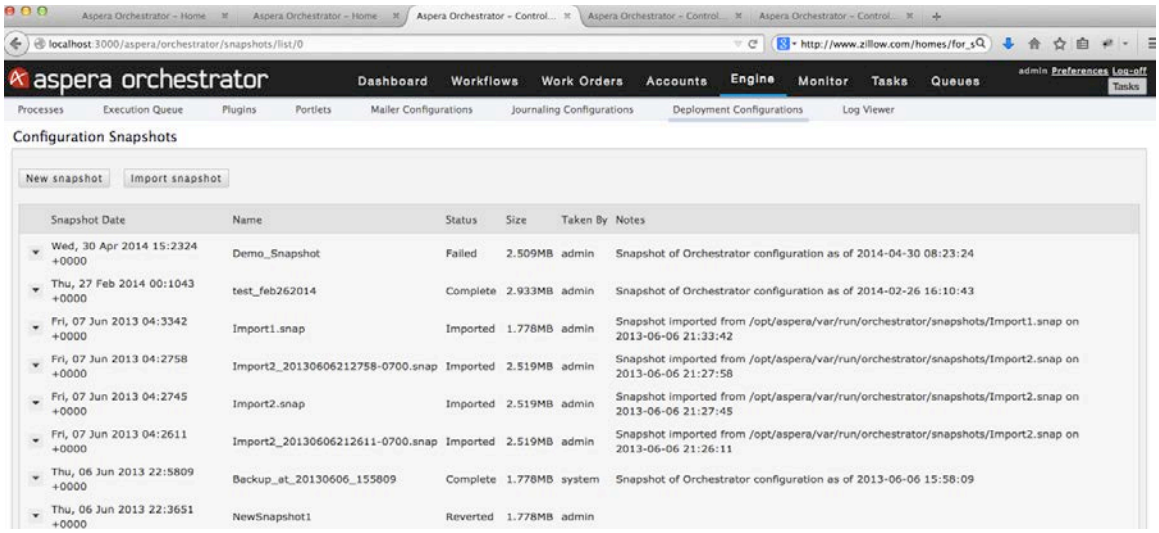
Orchestrator UI provides a utility to take a "snapshot" of Orchestrator. A snapshot is a backup of all your scripts, capsules, configuration files, plugins, plugin configurations, licenses and database tables (both table structure and table data). Snapshot functionality is commonly used for below use cases.

a) Take a regular backup of Orchestrator configurations.
b) Move Orchestrator configurations from one instance to another.

TAKE A NEW SNAPSHOT

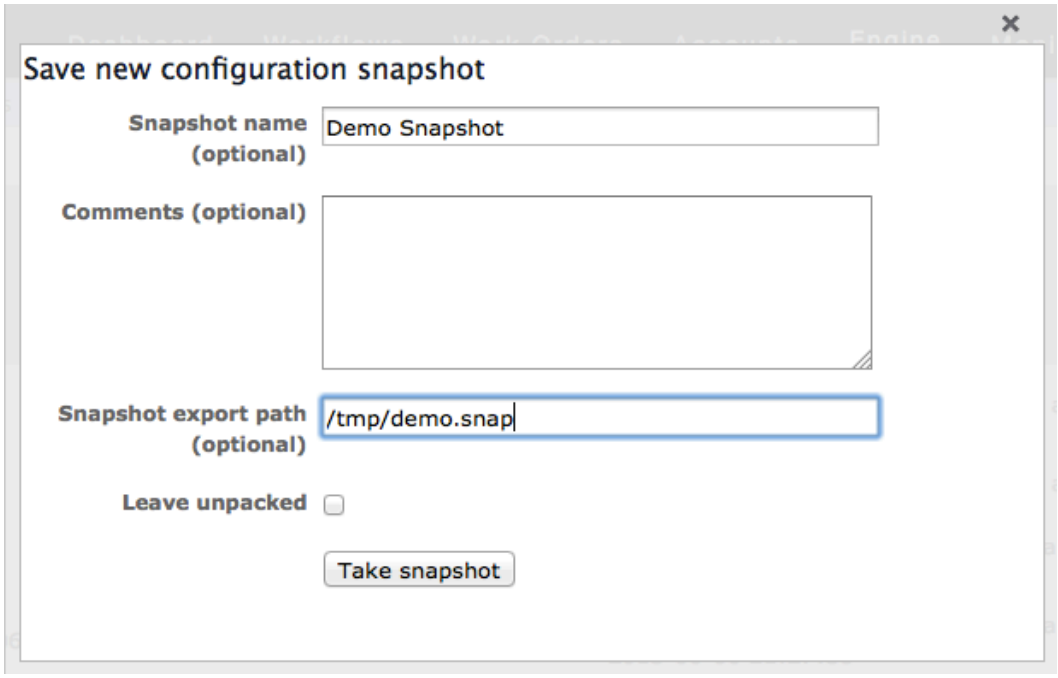
Below are the instructions on how to take a Snapshot.

Navigate to Engine tab and then to Deployment Configurations screen



Click on New snapshot button.

In the ensuing pop up, provide details to identify and store your snapshot.



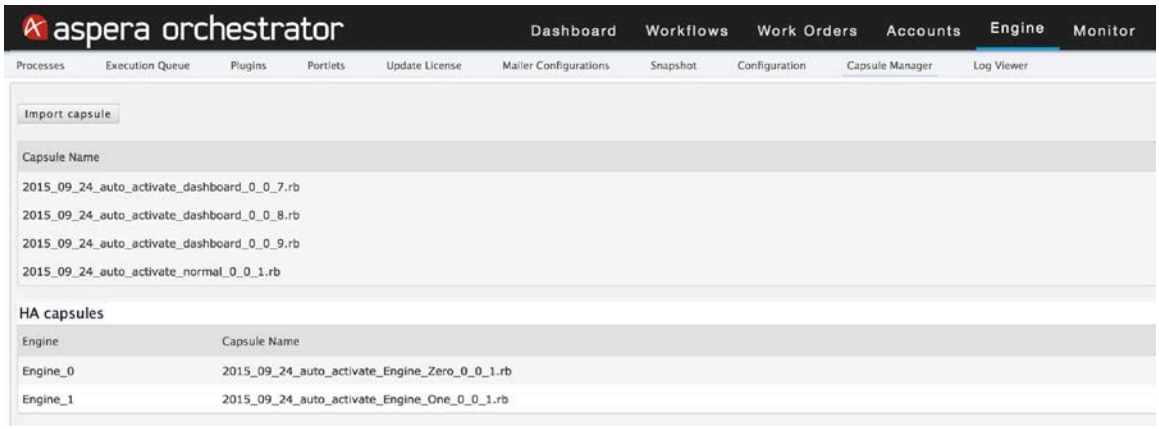
Click on Take Snapshot button to begin the snapshot process. The entire backup is compressed into a '.snap' file, which in this case is demo.snap located in /tmp. Default directory is the Orchestrator Archive directory (see Orchestrator installation directories section for exact location)

CAPSULE MANAGEMENT

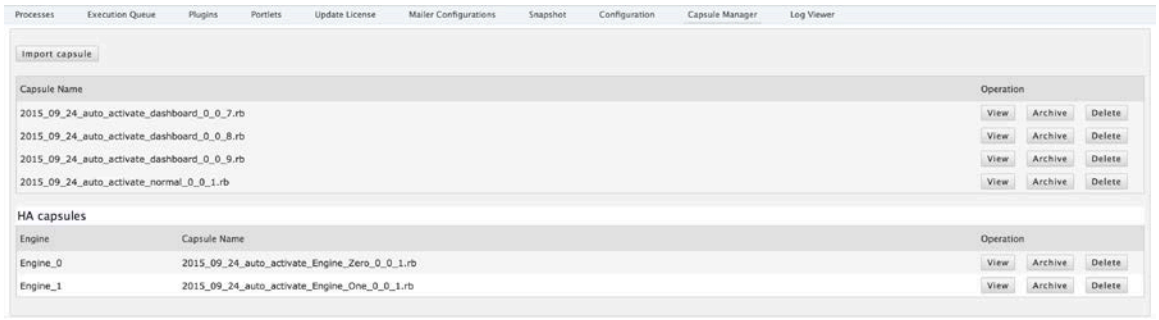
Capsules are orchestrator patches that are applied to add feature to fix bug without upgrading Orchestrator. Capsule code was incorporated on the orchestrator server by importing capsule from the Engine's menu for the Orchestrator v2.3.5 and earlier. From v2.4.1, buttons for importing of

capsules has moved to 'Capsule Manager'. Along with importing of capsule, existing capsules can be viewed, archived to the archive directory or deleted from the server.

Select the Engine menu and click on 'Capsule Manager'.



The screen lists all the capsules that are present on the Orchestrator server. Following operations can be performed on each of the capsule.



As show in the above screenshot, the capsules belonging to HA systems are visible as well.

In HA environment, there are multiple engines serving the Orchestrator server. Each engine can have its own capsule(s). HA capsules in the above screenshot shows one capsule from each engine instance '*Engine_0*' and '*Engine_1*'.

Same actions can be performed on the regular capsules and on HA capsules. They are discussed below

View - Clicking on the view button, opens up a modal window with the capsule code in the read only mode.

Archive – Clicking on the archive button will, prompt for confirmation to go ahead with archiving of the capsule.

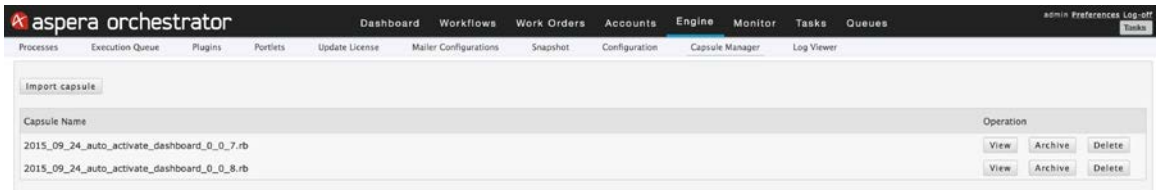
Note: Archiving a capsule, removes it from the capsule directory and moves it to archive directory. Orchestrator needs to be restarted to allow the change to take effect.

Delete – Deleting a capsule will remove the file from the Orchestrator server.

Note: Deleting a capsule, removes it from the Orchestrator server. This action is irreversible and Orchestrator needs to be restarted to allow the change to take effect.

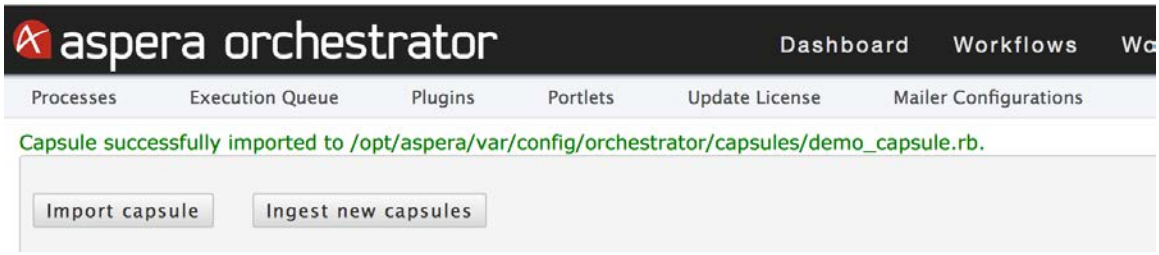
IMPORT CAPSULE

Capsule can be imported from the Orchestrator GUI by clicking on the 'Import capsule' button.



Clicking on it opens a Upload dialog box. Browse the '.rb' capsule file and click on Upload.

Upon successful import, following message should get displayed.



FIREWALL RULES

Aspera Products					
From	**To**	**Port**	**Protocol**	**Functionality**	**Direction**
Orchestrator	Enterprise server	40001	TCP	API Requests	Inbound to Enterprise server
Orchestrator	P2P Server	40001	TCP	API Requests	Inbound to P2P server
Orchestrator	Console	4406	TCP	Reporting	Inbound to Console
Orchestrator	Aspera Node API	9091, 9092	TCP	API Requests	Inbound to Aspera Nodes
Orchestrator	Aspera Faspex	443	TCP	API Requests	Inbound to Aspera Nodes
Orchestrator	Aspera Shares	443, 9092	TCP	API Requests	Inbound to Aspera Nodes

3rd party transcoders

From	To	Port	Protocol	Functionality	Direction
Orchestrator	Carbon Transcoder	1120, 1301	TCP	API Requests	Inbound to destination
Orchestrator	Rhozet Transcoder	8731	TCP	API Requests	Inbound to destination
Orchestrator	Elemental Transcoder	80, 443			Inbound to destination
Orchestrator	FlipFactory Transcoder	80, 443			Inbound to destination
Orchestrator	Vantage Transcoder	8676	TCP	API Requests	Inbound to destination
Orchestrator	Digital Rapids Stream	44000	TCP	Reporting	Inbound to destination
Orchestrator	Digital Rapids TM	44000			

3rd party QC tools					
From	**To**	**Port**	**Protocol**	**Functionality**	**Direction**
Orchestrator	Baton QC	8080	TCP	API Requests	Inbound to destination
Orchestrator	Aurora QC	1001	TCP	API Requests	Inbound to destination
Orchestrator	Cerify QC	80	TCP	Reporting	Inbound to destination

General tools					
From	**To**	**Port**	**Protocol**	**Functionality**	**Direction**
Orchestrator	Email relay	25	SMTP	Email	Inbound to destination
Orchestrator	Database	4406	TCP	MySQL queries	Inbound to destination
Orchestrator	Database	1433	TCP	MSSQL queries	Inbound to destination
Orchestrator	Database	1521	TCP	Oracle queries	Inbound to destination

Portlet					
From	**To**	**Port**	**Protocol**	**Functionality**	**Direction**
Aspera	Aspera Transfer	22 or	TCP	Authentication	Inbound to

connect client	server	33001			destination
Aspera connect client	Aspera Transfer server	33001	UDP	Transfer	Inbound to destination

CUSTOMIZE ORCHESTRATOR LOGO

Orchestrator ships with a default logo like below

A backup of the original aspera_orchestrator_logo.png should be made to revert the changes.

Customer's logo should overwrite the file at location - <Orchestrator_Install_Directory>/public/images/aspera_orchestrator_logo.png

Copy the same aspera_orchestrator_logo.png into the directory <Orchestrator Config Directory>/web/images/ so that the logo is preserved after upgrades.

Edit the file branding.yml in Orchestrator Config Directory

Change the below line to point to new image's path

```
$ logo: ../my/images/aspera_orchestrator_logo.png
```

MIGRATE FROM DISK PERSISTENCE TO DB PERSISTENCE

Trigger steps in Aspera Orchestrator store information in a persistent location to prevent duplicating triggers. Use the procedure below to migrate from disk (file) to db persistence.

IMPLEMENTATION PROCEDURE

1. Stop active work orders (by cancelling them, pausing them or destroying them) and any monitors that might start those work orders back.

2. Backup the configuration

```
$ cp <Orchestrator Config Directory>/orchestrator.yml <Orchestrator
Config Directory>/orchestrator.yml _<date>
```

3. Update the configuration by adding this line to the YML file.

```
persistence_mode: db
```

4. Stop Orchestrator (refer to the sections earlier in the document for details)

5. Migrate the persistence information from the prst files to the database.

```
$ cd <Orchestrator_Install_Directory>
$ ruby script/cmd
Dir.glob('<Orchestrator Run Directory>/*/*.prst').each { |prst|
    s=SharedState.retrieve_from_file(prst)
    SharedState.persist_to_db(s,prst)
```

```
    s_db=SharedState.retrieve_from_db(prst)
    if(s_db != s)
        throw "Error migrating SharedState file '#{prst}'"
    end
}
```

6. Start Orchestrator (refer to the sections earlier in the document for details)

7. Watch the orchestrator.log for errors and confirm that the workorders start successfully via the web interface.

ROLLBACK PROCEDURE

1. Backup the configuration

```
$ cp <Orchestrator Config Directory>/orchestrator.yml_<date>
<Orchestrator Config Directory>/orchestrator.yml
```

2. Update the configuration by removing this line from the YML file.

```
persistence_mode: disk
```

3. Stop Orchestrator (refer to the sections earlier in the document for details)

4. Start Orchestrator refer to the sections earlier in the document for details)

5. Watch orchestrator.log for errors and confirm that the workorders start successfully via the web interface.

ADD MONGREL PROCESSES FOR NEW APACHE PORT

Orchestrator Mongrels can be increased and have the port listen on a different Apache port. This use case is valid for users have a lot of API calls to Orchestrator that makes user interface slower.

Edit Orchestrator config (orchestrator.yml)

The Orchestrator config file, orchestrator.yml can be edited to increase number of mongrels. Find the entry max_mongrels and increase it to say 8.

Restart Orchestrator (see above section for more details on how to restart Orchestrator)

Create a Virtual Host for Orchestrator inside Apache's conf file

- Create a file in the /opt/aspera/common/apache/custom directory on Linux or C:\Progam Files (x86)\Aspera\Orchestrator\apache\custom\ in Windows. Edit the file to reflect below configuration. The name of the file can be custom_config.conf
- Edit Apache's conf file under /opt/aspera/common/apache/ashttpd.conf to check if custom directory is loaded
- Add below lines in the newly created file in the custom directory

```
Listen 4080
<VirtualHost *:4080>
      ServerAdmin webmaster@localhost


Alias /aspera/orchestrator/images
"/opt/aspera/orchestrator/public/images"
Alias /aspera/orchestrator/stylesheets
"/opt/aspera/orchestrator/public/stylesheets"
Alias /aspera/orchestrator/javascripts
"/opt/aspera/orchestrator/public/javascripts"
<Directory "/opt/aspera/orchestrator/public">
 Options -Indexes -FollowSymLinks
 AllowOverride none
 Order allow,deny
 Allow from all
</Directory>


<Proxy balancer://orchestrator_cluster>
 Allow from all
      BalancerMember http://127.0.0.1:3000
      BalancerMember http://127.0.0.1:3001
      BalancerMember http://127.0.0.1:3002
      BalancerMember http://127.0.0.1:3003
```

```
      BalancerMember http://127.0.0.1:3004
</Proxy>


ProxyPass /aspera/orchestrator/images !
ProxyPass /aspera/orchestrator/stylesheets !
ProxyPass /aspera/orchestrator/javascripts !


# send the proxy request


   # @web_root starts with / if not nil
   ProxyPass /aspera/orchestrator
balancer://orchestrator_cluster/aspera/orchestrator


   RequestHeader set X_FORWARDED_PROTO "https"


</VirtualHost>
```

Note: The number of mongrels running for Orchestrator must match the number BalancerMembers. Usage of port 4080 is just an example, please put in a value that is meaningful.

Repeat the above step for another port say 5080 for the rest of the ports 3005, 3006 and 3007

Restart Apache service using

```
# asctl apache:restart
```

Testing:

You should be able to reach Orchestrator at

- http://localhost:4080/aspera/orchestrator/
- Login with a valid user_id and password, you should be redirected to Orchestrator's Work Orders page

Replace localhost with the Machine IP if accessing from a different machine.

INCREASE APACHE PROXY TIMEOUT

Users with use cases that make Mongrels to execute longer queries experience proxy errors, as Apache is not receiving a response from the Mongrels. Below configuration increases the possible timeout values.

```
# tell Apache where to find static files /aspera/orchestrator
Alias /aspera/orchestrator/images
"/opt/aspera/orchestrator/public/images"
Alias /aspera/orchestrator/stylesheets
"/opt/aspera/orchestrator/public/stylesheets"
Alias /aspera/orchestrator/javascripts
"/opt/aspera/orchestrator/public/javascripts"
Alias /aspera/orchestrator/my/images
"/opt/aspera/var/config/orchestrator/web/images"
Alias /aspera/orchestrator/my/stylesheets
"/opt/aspera/var/config/orchestrator/web/stylesheets"
Alias /aspera/orchestrator/my/javascripts
"/opt/aspera/var/config/orchestrator/web/javascripts"
<Directory "/opt/aspera/orchestrator/public">
  Options -Indexes +FollowSymLinks
  AllowOverride none
  Order allow,deny
  Allow from all
</Directory>
<Directory "/opt/aspera/var/config/orchestrator/web">
  Options -Indexes -FollowSymLinks
  AllowOverride none
  Order allow,deny
  Allow from all
</Directory>

ProxyTimeout 1200

<Proxy balancer://orchestrator_cluster>
       BalancerMember http://127.0.0.1:3000
       BalancerMember http://127.0.0.1:3001
       BalancerMember http://127.0.0.1:3002
       BalancerMember http://127.0.0.1:3003
       BalancerMember http://127.0.0.1:3004
</Proxy>

# don't proxy static files, serve directoy
```

```
ProxyPass /aspera/orchestrator/images !
ProxyPass /aspera/orchestrator/stylesheets !
ProxyPass /aspera/orchestrator/javascripts !
ProxyPass /aspera/orchestrator/my/images !
ProxyPass /aspera/orchestrator/my/stylesheets !
ProxyPass /aspera/orchestrator/my/javascripts !


# send the proxy request


  # @web_root starts with / if not nil
ProxyPass /aspera/orchestrator
balancer://orchestrator_cluster/aspera/orchestrator timeout=1200
```

ASCTL AND ORCHESTRATOR

INTRODUCTION

Aspera's ASCTL functionality is compatible with Orchestrator v2.3 and Aspera Common greater than 1.12.12x. This will help users to control many important operations from command line.

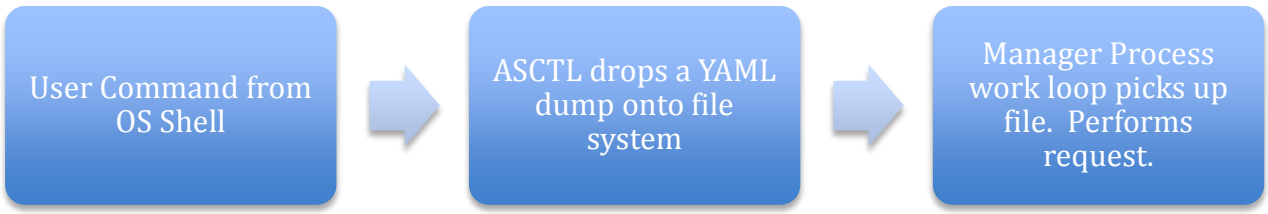
Currently Orchestrator provides command line utility for certain operations like start, stop and check status using the script "orchestrator". Other scripts exist to allow users to setup/install, backup data and migrate data.

The goal through ASCTL is to provide a single interface for all the tasks that can considerably reduce the effort needed for ASCTL to interactively take up Orchestrator tasks.

The methods, implementation, inputs and outputs along with the necessary screenshots are detailed in the section below.

DESIGN CONSIDERATIONS

Loading Rails will cause slow execution of the commands. Hence all the methods interacting with the Rails code dump a YAML file with instructions for the Orchestrator process to pick up and understand the task.

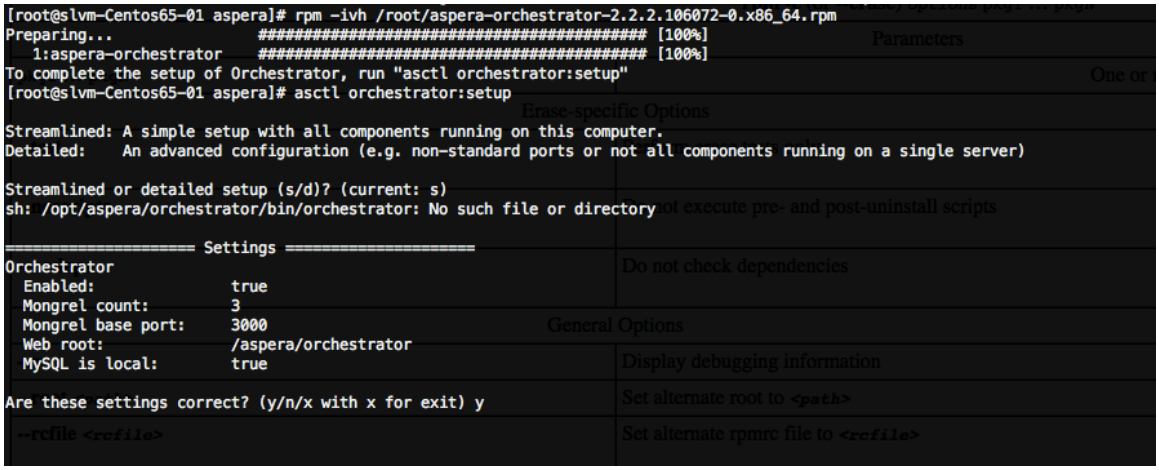


1. Diagram showing the interaction between OS Shell and Orchestrator

ASCTL ORCHESTRATOR METHODS – HIGH-LEVEL DESCRIPTION

ORCHESTRATOR SETUP

Method name	Setup
Command	asctl orchestrator:setup
Details	Configure this component
Inputs	None (Interactive)
Outputs	Interactively ask questions to setup Orchestrator. Display user inputs and begin install.



START ORCHESTRATOR

Method name	Start
Command	asctl orchestrator:start
Details	Used to instantiate orchestrator processes – manager, engine monitor, workers and mongrels.
Inputs	None
Outputs	STDOUT to show that Orchestrator processes have been stopped.

```
[root@slvm-Centos65-01 orchestrator]# asctl orchestrator:start
starting...
started...
nohup: redirecting stderr to stdout

Orchestrator Status:
  -> Orchestrator Manager running with pid: 10139
  -> Orchestrator Engine running with pid: 10022
  -> Mongrel serving orchestrator on port 3000 is running with pid: 10028
  -> Mongrel serving orchestrator on port 3001 is running with pid: 10031
  -> Mongrel serving orchestrator on port 3002 is running with pid: 10034
  -> Orchestrator Monitor running with pid: 10025
  -> Asynchronous Worker Process 0 is running with pid: 10039
  -> Asynchronous Worker Process 1 is running with pid: 10041
  -> Synchronous Worker Process 2 is running with pid: 10043
  -> Synchronous Worker Process 3 is running with pid: 10045
  -> Synchronous Worker Process 4 is running with pid: 10047
  -> Synchronous Worker Process 5 is running with pid: 10037
[root@slvm-Centos65-01 orchestrator]#
```

CHECK ORCHESTRATOR STATUS

Method name	Status
Command	asctl orchestrator:status
Details	Display the status: 'running', 'stopped', 'stop pending', etc.
Inputs	None
Outputs	STDOUT similar to below screenshot

STOP ORCHESTRATOR

Method name	Stop
Command	asctl orchestrator:stop
Details	Used to stop instantiated orchestrator processes – manager, engine monitor, workers and mongrels.
Inputs	None
Outputs	STDOUT to show that Orchestrator processes have been stopped.

```
[root@slvm-Centos65-01 orchestrator]# asctl orchestrator:stop
stopping...
stopped...

Orchestrator Status:
  -> Orchestrator Manager running with pid: 10139
  -> Orchestrator Engine NOT running
  -> Mongrel serving orchestrator on port 3000 is NOT running
  -> Mongrel serving orchestrator on port 3001 is NOT running
  -> Mongrel serving orchestrator on port 3002 is NOT running
  -> Orchestrator Monitor NOT running
  -> Asynchronous Worker Process 0 is NOT running
  -> Asynchronous Worker Process 1 is NOT running
  -> Synchronous Worker Process 2 is NOT running
  -> Synchronous Worker Process 3 is NOT running
  -> Synchronous Worker Process 4 is NOT running
  -> Synchronous Worker Process 5 is NOT running
[root@slvm-Centos65-01 orchestrator]#
```

ADMIN USER CONFIGURATION

Method name	configure_admin_user(login, email, password)
Command	asctl orchestrator:configure_admin_user login email [password]
Details	Create a new admin user (or modify the existing admin with the same name).
Inputs	Users' login (String), users' email (String) and password (String)
Outputs	STDOUT to indicate that the new admin has been created or an existing admin account has been modified

```
[root@slvm-Centos65-01 orchestrator]# asctl orchestrator:configure_admin_user "administrator" "admin@asperasoft.com" "PASSPhrase"
An admin user has complete control over Orchestrator processes and jobs, continue? (y/n default:n) y
Orchestrator: Configuration admin user... An install specific config file can be created at /opt/aspera/var/config/orchestrator/orchestrator.yml to overwrite defaults
Created admin user with login administrator
done
[root@slvm-Centos65-01 orchestrator]#
```

STOP ALL MONGRELS

Method name	stop_mongrels
Command	asctl orchestrator:stop_mongrels
Details	Stop the processes rendering the UI.
Inputs	None
Outputs	STDOUT to show that Orchestrator Mongrel processes have been stopped.

```
[root@slvm-Centos65-01 orchestrator]# asctl orchestrator:stop_mongrels
An install specific config file can be created at /opt/aspera/var/config/orchestrator/orchestrator.yml to overwrite defaults
Command stop_all executed.
```

START ALL MONGRELS

Method name	start
Command	asctl orchestrator:start_mongrels
Details	Start the processes rendering the UI.
Inputs	None
Outputs	STDOUT to show that Orchestrator Mongrel processes have started.

```
[root@slvm-Centos65-01 orchestrator]# asctl orchestrator:start_mongrels
An install specific config file can be created at /opt/aspera/var/config/orchestrator/orchestrator.yml to overwrite defaults
Command start_all executed.
[root@slvm-Centos65-01 orchestrator]#
```

ASYNCHRONOUS WORKER COUNT

Method name	asynchronous_worker_count
Command	asctl orchestrator:asynchronous_worker_count [count]
Details	Display the number of workers executing asynchronous steps
Inputs	Is_Synchronous (Boolean), Worker count (Integer)
Outputs	STDOUT to indicate the worker count. Or if the worker count has been modified

SYNCHRONOUS WORKER COUNT

Method name	worker_count(true, count)
Command	asctl orchestrator:synchronous_worker_count [count]
Implemented?	No
Details	Display the number of workers executing synchronous steps
Inputs	Is_Synchronous (Boolean), Worker count (Integer)
Outputs	STDOUT to indicate the worker count. Or if the worker count has been modified

BASE PORT

Method name	base_port(port)
Command	asctl orchestrator:base_port
Details	Display lowest port for the Mongrel instances.
Inputs	Port number (Integer)
Outputs	STDOUT to display current base port. If a port is provided, replace the current base port and display success message.

CREATE SETUP FILE

Method name	create_setup_file(file_path)
Command	asctl orchestrator:create_setup_file file_path
Details	Create a reusable file that contains answers to the setup questions.
Inputs	File path (String)
Outputs	STDOUT to indicate the progress of the snapshot and eventually a success or failure message

```
[root@slvm-Centos65-01 orchestrator]# asctl orchestrator:create_setup_file "/tmp/setup.txt"

Streamlined: A simple setup with all components running on this computer.
Detailed:    An advanced configuration (e.g. non-standard ports or not all components running on a single server)

Streamlined or detailed setup (s/d)? (current: s) s

Orchestrator
  What base port should the Mongrel servers start at? (current: 3000)

  ERROR: Port number 3000 which falls within the range '3000' to '3002' is in use.

  What base port should the Mongrel servers start at? (current: 3000) 3003

==================== Settings ====================
Orchestrator
  Enabled:              true
  Mongrel count:        3
  Mongrel base port:    3003
  Web root:             /aspera/orchestrator
  MySQL is local:       true

Saved to file '/tmp/setup.txt'
[root@slvm-Centos65-01 orchestrator]#
```

CREATE SNAPSHOT

Method name	snapshot
Command	asctl orchestrator:create_snapshot

Details	Create snapshot of all Orchestrator configuration and design-time data.
Inputs	None
Outputs	STDOUT to indicate the progress of the snapshot and eventually a success or failure message

```
[root@slvm-Centos65-01 orchestrator]# asctl orchestrator:create_snapshot
Orchestrator: Create snapshot... Snapshot of Orchestrator configuration was successfully created at 2015-04-22T13:45:25-07:00.
Snapshot file located at /opt/aspera/var/archive/orchestrator/snapshots/Backup_at_20150422_134524.snap
done
[root@slvm-Centos65-01 orchestrator]#
```

DISABLE ORCHESTRATOR

Method name	disable
Command	asctl orchestrator:disable
Details	Disables Orchestrator; removes entry from /etc/init.d/ and /etc/rc.d/
Inputs	None
Outputs	STDOUT to indicate Orchestrator is disabled.

```
[root@slvm-Centos65-01 orchestrator]# asctl orchestrator:disable
Orchestrator: Unregister with Apache... done
Orchestrator: Uninstall service... done
Orchestrator: Disable... done
[root@slvm-Centos65-01 orchestrator]#
```

DATABASE BACKUP

If no file name provided, it will default the output to the path "/opt/aspera/var/archive/orchestrator/databases/config" with the conf_"timestamp".sql as file name.

Method name	backup_database(path)
Command	asctl orchestrator:backup_database path
Details	Back up the orchestrator database (full backup).
Inputs	Path (String)
Outputs	STDOUT to indicate the database has been backed up. All Orchestrator backups are stored under Orchestrator Archive Directory (see section Orchestrator installation directories)

GENERATE CONFIG

Method name	generate_config
Command	asctl orchestrator:generate_config
Details	Regenerate the configuration files using the current settings
Inputs	None
Outputs	STDOUT to indicate the configuration file has been generated

HELP

Method name	help
Command	asctl orchestrator:help
Details	Describe the commands and usage
Inputs	None
Outputs	STDOUT the text describing the commands and usage

```
[root@slvm-Centos65-01 orchestrator]# asctl orchestrator:help
Orchestrator: Orchestrator ASCTL help...
Orchestrator Commands (prefix with 'orchestrator:')


asynchronous_worker_count       To view (Or change) the asynchronous thread count
backup_database                 Back up the orchestrator database(full backup) into a dump file.
base_port                       Display/change lowest port for the Mongrel instances.
change_configuration_parameter  Modify a configuration parameter
configure_admin_user            To Configure the setup of a admin user
create_setup_file               Create a reusable file that contains answers to the setup questions.
create_snapshot                 Create snapshot of all Orchestrator configuration and design-time data.
disable                         Disables Orchestrator
export_plugin                   To export an existing plugin from the Orchestrator
export_workflow                 To export an existing workflow from the Orchestrator
generate_config                 Regenerate the configuration files using the current settings
generate_config_file            Generate the config file
help                            Describe the commands and usage
info                            Print configuration info about this component
list_configuration              List all modifiable configuration parameters
list_workflows                  List all workflows designed in this server
migrate_database                Update database to latest schema.
mongrel_count                   Display the current count of mongrels configured to run.
rake                            Invoke a rake command
rake_command                    Execute a rake command
restart                         Restart Orchestrator
restore_database                Restores mysql from the dump file provided
restore_snapshot                Restore Orchestrator configuration and design time data from a snapshot.
setup                           Configure the Orchestrator component
setup_from_file                 Configure the Orchestrator using the given file settings
start                           To start the Orchestrator engine
status                          Shows the current status of the Orchestrator
stop                            To stop the Orchestrator engine
switch_orchestrator_version     To switch the Orchestrator version
synchronous_worker_count        To view (Or change) the synchronous thread count
upgrade                         To upgrade the Orchestrator to a new version
uri_namespace                   Display/Change the URI namespace
version                         Display Orchestrator Version
web_root                        Display/Change the web root of Orchestrator
stop_mongrels                   Stop the processes rendering the UI
start_mongrels                  Start the processes rendering the UI
restore_db_from_file            Restores mysql from the dump file provided
done
[root@slvm-Centos65-01 orchestrator]#
```

INFO

Method name	info
Command	asctl orchestrator:info
Details	Print configuration info about this component
Inputs	None
Outputs	STDOUT the configuration info about Orchestrator

```
[root@slvm-Centos65-01 orchestrator]# asctl orchestrator:info
Orchestrator
  status:
Orchestrator Status:
  -> Orchestrator Manager running with pid: 10139
  -> Orchestrator Engine running with pid: 11086
  -> Mongrel serving orchestrator on port 3001 is running with pid: 23434
  -> Mongrel serving orchestrator on port 3002 is running with pid: 23437
  -> Mongrel serving orchestrator on port 3003 is NOT running
  -> Extra Mongrel serving orchestrator on port 3000 is running with pid: 23431
  -> Orchestrator Monitor running with pid: 11089
  -> Asynchronous Worker Process 0 is running with pid: 11103
  -> Asynchronous Worker Process 1 is running with pid: 11105
  -> Asynchronous Worker Process 2 is running with pid: 11107
  -> Synchronous Worker Process 3 is running with pid: 11109
  -> Synchronous Worker Process 4 is running with pid: 11111
  -> Synchronous Worker Process 5 is running with pid: 11101
  enabled:          true
  uri_namespace:    /aspera/orchestrator
  mongrel_count:    3
  base_port:        3000
[root@slvm-Centos65-01 orchestrator]#
```

LIST CONFIGURATION

Method name	list_configuration_parameters
Command	asctl orchestrator:list_configuration_parameters
Details	List all modifiable configuration parameters
Inputs	None
Outputs	STDOUT the modifiable configuration parameters

```
[root@slvm-Centos65-01 orchestrator]# asctl orchestrator:list_configuration_parameters
Orchestrator: List configuration parameters...

rails_log_level               :                 error
log_level                     :                 debug
journal_on                    :                 true
asynch_threads                :                 2
engine_heartbeat              :                 1
date_format                   :                 american
custom_tables                 :
max_tasks_displayed           :                 12
log_retention_days            :                 7
install_images_dir            :                 /opt/aspera/var/config/orchestrator/images
install_config_file           :                 /opt/aspera/var/config/orchestrator/orchestrator.yml
fast_start                    :                 false
active_active                 :                 false
install_css_dir               :                 /opt/aspera/var/config/orchestrator/stylesheets
mongrel_starting_port         :                 3000
journal_kept_duration         :                 10
referer_hosts                 :
ad_user_groups                :                 Contributor
load_basic_workflow_dir       :                 /opt/aspera/orchestrator/vendor/workflows
log_file                      :                 /opt/aspera/var/run/orchestrator/log/orchestrator.log
run_dir                       :                 /opt/aspera/var/run/orchestrator
purge_cutoff                  :                 60
use_db_time                   :                 false
orchestrator_dir              :                 /opt/aspera/orchestrator
persistence_mode              :                 disk
mongrel_action_timeout        :                 300
hide_preferences_for_groups   :
personalized                  :                 true
install_pages_dir             :                 /opt/aspera/var/config/orchestrator/pages
archive_dir                   :                 /opt/aspera/var/archive/orchestrator
default_step_timeout          :                 60
synch_threads                 :                 4
build                         :                 3
db_bin                        :                 /opt/aspera/common/mysql/bin
publish_default_workflows     :                 true
use_new_designer              :                 true
web_root                      :                 /aspera/orchestrator
disable_autocomplete_login_page:                        false
config_dir                    :                 /opt/aspera/var/config/orchestrator
aspera_dir                    :                 /opt/aspera
cleanup_cutoff                :                 30
```

LIST WORKFLOWS

Method name	list_workflows(workflow_folder, status_filter, format)
Command	asctl orchestrator:list_workflows [workflow_folder] [status_filter] [format]
Details	List all workflows designed in this server
Inputs	Workflow_folder (String), Status_Filter (String), Format (String)
Outputs	STDOUT the workflows in this server. If a workflow folder is specified, show workflows inside that folder only. Provide options to output an XML file/string or JSON String

```
[root@slvm-Centos65-01 orchestrator]# asctl orchestrator:list_workflows
Orchestrator: List workflows...
WorkFlow List(Workflow names along with their ids)
"Sub_Transfer_to_S3"                                    4
"ADI Ingest"                                            2
"111"                                                   7
"Media validation and virus scan  (import_1)"           5
"Demo_to_bell"                                          6
"Ingest_Demo"                                           1
done
[root@slvm-Centos65-01 orchestrator]#
```

MIGRATE DATABASE

Method name	migrate_database
Command	asctl orchestrator:migrate_database
Details	Update database to latest schema.
Inputs	None
Outputs	STDOUT the result of the operation

MONGREL COUNT

Method name	mongrel_count(number)
Command	asctl orchestrator:mongrel_count [number]
Details	Display the current count of mongrels configured to run.
Inputs	None
Outputs	STDOUT the result of the operation

RAKE COMMANDS

Method name	rake_command
Command	asctl orchestrator:rake [arguments]
Implemented?	No
Details	Invoke a rake command. The arguments are determined by the command executed.
Inputs	None
Outputs	STDOUT the result of the operation

RESTART ORCHESTRATOR

Method name	restart
Command	asctl orchestrator:restart
Implemented?	Yes.
Details	Restart Orchestrator
Inputs	None
Outputs	STDOUT the stop operation, followed by the start operation

RESTORE FROM DATABASE FILE

Method name	restore_database(filepath)
Command	asctl orchestrator:restore_database [filepath]
Implemented?	Yes.
Details	If a mysql dump file which contains an Orchestrator database is provided, the current database will be backed up and this database will be used as primary
Inputs	Filepath (String)
Outputs	STDOUT the progress of the operation

```
[root@slvm-Centos65-01 orchestrator]# asctl orchestrator:restore_database "/tmp/sample.sql"
This will destroy all data in the current Orchestrator database, continue? (y/n default:n) y
Orchestrator: Restore database (this may take a while)... false
done
[root@slvm-Centos65-01 orchestrator]#
```

RESTORE FROM SNAPSHOT FILE

Method name	restore_snapshot(filepath)
Command	asctl orchestrator:restore_snapshot [filepath]
Details	Restore Orchestrator configuration and design time data from a snapshot. If a orchestrator snapshot file is provided, the current configuration will be backed up and the instance will be reverted to the supplied snapshot file
Inputs	Filepath (String)
Outputs	STDOUT the progress of the operation

SETUP ORCHESTRATOR FROM FILE

Method name	setup_from_file (filepath)
Command	asctl orchestrator:setup_from_file [filepath]
Details	Run setup using the answers from a file created using "create_setup_file". Redo the setup of Orchestrator configuration to parameters from this setup file.
Inputs	Filepath (String)
Outputs	STDOUT the progress of the operation

```
[root@slvm-Centos65-01 orchestrator]# asctl orchestrator:setup_from_file "/tmp/setup.txt"

Orchestrator
  What base port should the Mongrel servers start at? (current: 3003) 3009

==================== Settings ====================
Orchestrator
  Enabled:               true
  Mongrel count:         3
  Mongrel base port:     3009
  Web root:              /aspera/orchestrator
  MySQL is local:        true

Are these settings correct? (y/n/x with x for exit) y
Orchestrator
  Enabling orchestrator...

--Generating Orchestrator config file
--Setting up database configuration file
(in /opt/aspera/orchestrator-2.2.2.105007)
orchestrator already exists
(in /opt/aspera/orchestrator-2.2.2.105007)
--Generating Orchestrator executables
--Setting up Orchestrator service
/sbin/chkconfig
--Setting up install specific images
--Setting up install specific stylesheets
--Setting up install specific pages
--Setting up Apache as a proxy
Enable Apache proxying for Apache (from Aspera common)
Updating Apache configuration file /opt/aspera/common/apache/custom/orchestrator_orchestrator.conf
done
  Task Name                                                Status
============================================================
Orchestrator
  Enabling orchestrator                                      OK

[root@slvm-Centos65-01 orchestrator]#
```

ORCHESTRATOR WEBROOT (URI NAMESPACE)

Method name	uri_namespace (namespace)
Command	asctl orchestrator:uri_namespace [namespace]
Details	Configure or change Orchestrator web root parameter
Inputs	Namespace (String)
Outputs	STDOUT the result of the operation

```
[root@slvm-Centos65-01 orchestrator]# asctl orchestrator:uri_namespace
Orchestrator: /aspera/orchestrator
[root@slvm-Centos65-01 orchestrator]#
```

ORCHESTRATOR VERSION

Method name	version
Command	asctl orchestrator:version
Details	Display the currently set up version.
Inputs	Filepath (String)
Outputs	STDOUT the version of Orchestrator

ORCHESTRATOR EXTERNAL INTEGRATION API

INTRODUCTION

Orchestrator exposes web services end points to allow 3rd party applications to programmatically control some of its functions such as launching and monitoring work-orders.

The API is REST based. Users can use HTTP or HTTPS as the transport protocol.

The HTTP GET method is supported for all requests. The HTTP POST method is also supported to initiate a work order with some user-defined data in the POST body (e.g. XML data).

The status code in the HTTP responses is 200 for success and 400 and above for all types of errors.

The data format in the responses can be XML or JSON. The response format can be controlled by submitting a parameter 'format' with value as 'xml' or 'json' or by adding extensions to the 'id'. Read the sections below for more detailed examples.

NOTE TO DEVELOPERS

Beginning with Orchestrator v 2.3.5 the API URL to access Orchestrator API has changed. The change is backward compatible, meaning the old URL containing the word "workflow_reporter" is still operational but new API calls will be available ONLY via the new API URL.

Also, beginning with Orchestrator v 2.3.5 API document, emphasis is put on using secure means of login – such as HTTP Basic authentication or using API key instead of password or using a scrambled password instead of password. Providing user name and password "inline" is highly discouraged.

WHAT'S NEW IN THE ORCHESTRATOR 2.4.1 API DOCUMENT

As mentioned above, a major change introduced in Orchestrator v2.3.5 and continued in Orchestrator v2.4.1 is the URL that exposes the Orchestrator API methods. In addition, now all API calls have JSON and XML response support. In case of failure scenarios, HTTP error codes such as 400 (Bad request), 401 (Access Denied), 403 (Authentication failure) and 404 (Method not found)

AUTHENTICATION

API calls require proper authentication and are subject to the SAME authorization requirements as direct access to the application through its web based user interface.

There are a number of ways to authenticate API calls.

Authentication Model	Description
HTTP Basic Authentication (most secure)	HTTP basic authentication is supported and is recommended to developers as the primary authentication model. The advantage of this method is credentials are within each request but they are encoded versus being in clear. Username and password are combined into a string "username:password", then they are Base64 encoded and put in the HTTP Authorization header such as: Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
API key or Scrambled password	For every Orchestrator user an API key is generated and so is a scrambled password. The values can be found in the User details page. Either of these values can be used in the REST URLs for authentication
Application Cookie	Another way is to assume that the client is able to maintain state information (cookie style). Once the authentication is made, further calls don't need to set the credentials. The hostname in the URLs is localhost in this document to represent the IP address or the fully qualified hostname of your Orchestrator node. Example cookie style request: http://localhost/aspera/orchestrator/api/authenticate/login=admin?password=aspera After a successful authentication, the next URL (e.g. initiate work order) can be called without providing the login and password parameters, until the user is logged off.
Login and Password (least secure)	Deprecated in future releases By inline, we mean each HTTP URL needs to include the authentication information (typically login/password). http://<Orchestrator_IP_address>/aspera/orchestrator/api/processes_status/0?login=<account_name>&password=<account_password> Such as: http://localhost/aspera/orchestrator/api/processes_status/0?login=admin&password=aspera

The account specified in the login section of the URL must be part of the Orchestrator Operator group and the account must have run permission on the workflow(s) for the API invocations like initiate a work order, fetch output spec of a work order and fetch output of a work order to succeed.

Note: *A combination of username/password - admin/aspera - is used in the rest of this document as an example for Orchestrator login account/password.*

Sample returned XML on success:

```
<?xml version="1.0"?>
<user time="2011-10-20 17:47:05
UTC" action="authenticate" id="1" login="admin"/>
```

Sample returned XML on bad password case:

```
<?xml version="1.0"?>
<error time="2011-10-20 17:45:39 UTC" action="status"
id="0">#<RuntimeError: Authentication failed></error>
```

Sample returned XML on user does not exist case:

```
<?xml version="1.0"?>
<error time="2011-10-20 17:48:09 UTC" action="status"
id="0">#<RuntimeError: Authentication failed></error>
```

To log off, use this URL:

http://localhost/aspera/orchestrator/users/logoff/xml

Sample returned XML on logoff:

```
<?xml version="1.0"?>
<user time="2012-01-25 06:38:22 UTC" action="logoff" id="1"
login="admin"/>
```

WORKFLOW RELATED API CALLS

LIST AVAILABLE WORKFLOWS ON THE SYSTEM

Description:

A method to enumerate all the Workflows in Orchestrator and list where the workflow is in a published state or not. Only a published workflow can be initiated as a work order.

Usage:

For an authenticated user:

http://<Orchestrator_IP_address>/aspera/orchestrator/api/workflows_list/0

With inline authentication:

http://<Orchestrator_IP_address>/aspera/orchestrator/api/workflows_list/0?login=admin&password=aspera

Note: A work order can only be instantiated for published workflows.

Sample returned XML:

```
<?xml version="1.0"?>
<workflows time="2011-10-20 17:18:58 UTC" action="list" id="0">
    <workflow id="2"
              name="Incoming File Scanning Daemon"
              published_status="published"
              published_revision_id="13"
              latest_revision_id="15"
              last_modification="2011-04-12 06:56:16 UTC" >
    </workflow>
    <workflow id="3"
              name="Incoming Clean File Staging Daemon"
              published_status="published"
              published_revision_id="9"
              latest_revision_id="9"
              last_modification="2011-03-30 19:47:38 UTC" >
        Created from revision #4 from workflow 'Incoming File Scanning
Daemon'
    </workflow>
    <workflow id="4"
              name="Asset and Schedule Import Daemon"
              published_status="draft"
              published_revision_id="-"
              latest_revision_id="20"
              last_modification="2011-05-06 19:20:36 UTC" >
    </workflow>
</workflows>
```

Sample returned JSON:

```
{
   "workflows":{
      "id":0,
      "action":"list",
      "workflow":[
         {
            "last_modification":"2015-12-15T00:21:08Z",
            "published_status":"published",
            "published_revision_id":4,
            "id":1,
            "latest_revision_id":4,
            "full_name":"TestFol1: fanout test",
            "comments":"",
            "portable_id":"fanout_test",
            "name":"fanout test",
            "folder_id":36
         },
         {
            "last_modification":"2015-12-15T00:21:08Z",
            "published_status":"published",
            "published_revision_id":1,
            "id":2,
            "latest_revision_id":1,
            "full_name":"TestFol1: send email",
            "comments":"",
            "portable_id":"send_email",
            "name":"send email",
            "folder_id":36
         }
      ],
      "time":"2016-01-11 20:33:05"
   }
}
```

FETCH INPUTS SPECIFICATION FOR A WORKFLOW

Description:

An API method to obtain the list of runtime parameters along with a flag indicating whether the parameter is required or optional.

Usage:

http://<Orchestrator_IP_address>/aspera/orchestrator/api/workflow_inputs_spec/158?login=admin&password=aspera

The required input parameter is:

- workflow_reporter/workflow_inputs_spec/<workflow_id> : specifies the workflow id

Sample returned XML:

```
<?xml version="1.0"?>
<workflow_inputs_spec time="2015-04-30 16:19:21 UTC"
action="inputs_spec">
  <workflow_name>ADI Ingest</workflow_name>
  <workflow_id>158</workflow_id>
      <parameter id="288" name="md5_check">
        <workflow_id>158</workflow_id>
        <value_type>flag</value_type>
        <optional>false</optional>
        <value></value>
      </parameter>

      <parameter id="290" name="filesize_check">
        <workflow_id>158</workflow_id>
        <value_type>flag</value_type>
        <optional>false</optional>
        <value></value>
      </parameter>

      <parameter id="289" name="dtd_check">
        <workflow_id>158</workflow_id>
        <value_type>flag</value_type>
        <optional>false</optional>
        <value></value>
      </parameter>
</workflow_inputs_spec>
```

Sample returned JSON:

```
{
   "workflow_inputs_spec":{
      "workflow_id":122,
      "time":"2016-01-12 19:11:57",
      "parameter":[
         {
```

```
            "name":"path_to_xml",
            "value_type":"string",
            "optional":true,
            "value":"undefi",
            "id":246
         },
         {
            "name":"asset_path",
            "value_type":"string",
            "optional":true,
            "value":"undef",
            "id":247
         }
      ],
      "action":"inputs_spec",
      "workflow_name":"Sub_Fileprep_action"
   }
}{
   "workflow_inputs_spec":{
      "workflow_id":1,
      "time":"2016-01-12 19:13:20",
      "parameter":[

      ],
      "action":"inputs_spec",
      "workflow_name":"fanout test"
   }
}
```

CHECK THE RUNNING STATUS FOR ALL WORKFLOWS

Description:

An API method to indicate which workflows have running work-orders and which are not being run.

Usage:

http://<Orchestrator_IP_address>/aspera/orchestrator/api/workflows_status/0?login=admin&password=aspera

Sample returned XML:

```
<?xml version="1.0"?>
<workflows time="2012-01-25 07:22:13 UTC" action="status" id="0">
    <workflow id="1" name="AAA - file processing for client"
status="not running" last_activity="-"/>
    <workflow id="3" name="AAA - retrieve file for distribution"
status="not running" last_activity="-"/>
    <workflow id="4" name="AAA - Master Controller processing"
status="not running" last_activity="-"/>
    <workflow id="6" name="AAA - Transcoding" status="not running"
last_activity="-"/>
    <workflow id="10" name="test faspex" status="not running"
last_activity="-"/>
    <workflow id="13" name="test mapping" status="running"
last_activity="2012-01-25 07:21:23 UTC"/>
    <workflow id="14" name="test multi role" status="running"
last_activity="2012-01-23 20:11:37 UTC"/>
</workflows>
```

Sample returned JSON:

```
{
   "workflows":{
      "action":"status",
      "id":"13",
      "workflow":{
         "id":"13",
         "statuses":[
            {
               "count":"614",
               "workflowName":"label workflow",
               "status":"Canceled",
               "last_activity":"2016-01-04 23:07:29"
            },
            {
               "count":"493",
               "workflowName":"label workflow",
               "status":"Complete",
               "last_activity":"2016-01-08 01:08:38"
            },
            {
               "count":"198",
               "workflowName":"label workflow",
               "status":"Error",
```

```
                "last_activity":"2016-01-04 23:04:13"
            },
            {
                "count":"4",
                "workflowName":"label workflow",
                "status":"Failed",
                "last_activity":"2015-12-22 23:26:40"
            },
            {
                "count":"1",
                "workflowName":"label workflow",
                "status":"In Progress",
                "last_activity":"2016-01-08 01:08:30"
            }
        ],
        "name":"label workflow"
    },
    "time":"2016-01-11 19:55:24"
  }
}
```

CHECK THE RUNNING STATUS FOR A SPECIFIC WORKFLOW

Description:

A method to indicate if a specific workflow identified by its workflow ID is currently running (i.e. some work orders have been instantiated and are currently being executed) or not.

Usage:

http://<Orchestrator_IP_address>/aspera/orchestrator/api/workflows_status/<workflow_id>?login=admin&password=aspera

Example:

http://localhost/aspera/orchestrator/api/workflows_status/1?login=admin&password=aspera

The required input parameter is:

- workflow_reporter/workflows_status/<workflow_id> : specifies the workflow id

Sample returned XML:

```
<?xml version="1.0"?>
<workflows time="2012-01-25 07:31:17 UTC" action="status" id="1">
    <workflow id="1" name="AAA - file processing for client"
status="not running" last_activity="-"/>
</workflows>
```

CHECK THE DETAILED RUNNING STATUS FOR A SPECIFIC WORKFLOW

Description:

A method to indicate for a given workflow the count of work orders in each execution status.

Usage:

http://<Orchestrator_IP_address>/aspera/orchestrator/api/workflow_details/<workflow_id>?login=admin&password=aspera

Example:

http://localhost/aspera/orchestrator/api/workflow_details/13?login=admin&password=aspera

The required input parameter is:

- workflow_reporter/workflows_details/<workflow_id> : specifies the workflow id

Sample returned XML:

```
<?xml version="1.0"?>
<workflows time="2012-01-25 07:11:52 UTC" action="details"  id="13">
  <workflow id="13" name="test mapping" >
```

```
      <status type="Complete" count="1" last_activity="2012-01-23
19:11:33"/>
      <status type="Failed" count="1764" last_activity="2012-01-25
07:11:15"/>
      <status type="In Progress" count="2" last_activity="2012-01-25
07:11:29"/>
      <status type="Error" count="1" last_activity="2012-01-23
19:05:51"/>
  </workflow>
</workflows>
```

Sample returned JSON:

```
{
   "workflows":{
      "action":"status",
      "id":"13",
      "workflow":{
         "id":"13",
         "statuses":[
            {
               "count":"614",
               "workflowName":"label workflow",
               "status":"Canceled",
               "last_activity":"2016-01-04 23:07:29"
            },
            {
               "count":"493",
               "workflowName":"label workflow",
               "status":"Complete",
               "last_activity":"2016-01-08 01:08:38"
            },
            {
               "count":"198",
               "workflowName":"label workflow",
               "status":"Error",
               "last_activity":"2016-01-04 23:04:13"
            },
            {
               "count":"4",
               "workflowName":"label workflow",
               "status":"Failed",
               "last_activity":"2015-12-22 23:26:40"
            },
            {
               "count":"1",
               "workflowName":"label workflow",
               "status":"In Progress",
               "last_activity":"2016-01-08 01:08:30"
            }
         ],
         "name":"label workflow"
      },
      "time":"2016-01-11 19:55:24"
   }
}
```

Note: Only the statuses that are encountered are reported. So for example, if no work-orders have failed for that workflow, no lines will be reported for that status.

WORK ORDER-RELATED API CALLS

INITIATE A WORK ORDER

Description:

This API method provides a way to initiate a new work order for a given workflow. If the call is successful, the created work order id is returned. It can be used to subsequently track the execution status of that work order. Note that if this workflow has sub-workflows, they are not accessible for monitoring via the API as only the parent work order id is returned.

The request can be either asynchronous or synchronous. If asynchronous, it will return as soon as the work order is created. If synchronous, it will return after a specified step in the workflow is executed.

The response can be either in XML or JSON format.

As mentioned above, the account specified in the login section of the URL must be part of the Orchestrator Operator group and the account must have run permission on the workflow.

If the workflow has run-time inputs, they can be specified in the URL with the notation external_parameters[<name_of run-time input-n>]=<value of run-time input-n>

Usage for an asynchronous request with XML response:

http://<Orchestrator_IP_address>/aspera/orchestrator/api/initiate/xml?login=admin&password=aspera&work_order[workflow_id]=<workflow id>&external_parameters[<name_of run-time input1>]=<value of run-time input1>&external_parameters[<name_of run-time input-n>]=<value of run-time input-n>...

The required input parameter is:

- work_order[workflow_id]=<integer> : specifies the workflow id as can be obtained in Orchestrator workflow editor

The optional input parameters are:

- work_order[higherPriority]=on : indicates that the work order is launched with a dedicated worker process.
- preemptive_level= <integer> : indicates a preemptive priority with the specified integer as the level of preemption (lowest go first)

- work_order[priority]=<integer from 1 → low to 3 → high> : specifies the priority level for that work order.
- work_order[name]=<string> : specifies a name for the work order replacing the generated default name that is based on the workflow name.
- external_parameters[<param_name>]=<string> : specifies a name for a parameter to be passed as run-time input to the workflow. <param_name> must match the name defined in the workflow as run-time input. Multiple external parameters can be passed all separated by the character '&'

Example:

http://localhost/aspera/orchestrator/api/initiate/xml?login=admin&password=aspera&work_order[workflow_id]=1

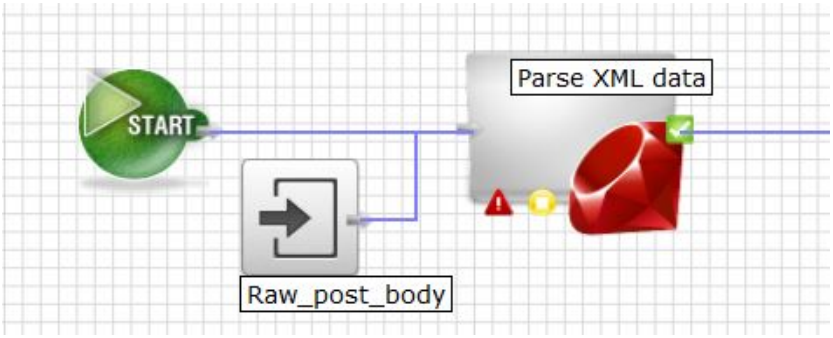
Sample returned XML:

```
<?xml version="1.0"?>
<work_order time="2014-11-25 13:24:01 UTC" action="initiate" id="8464"
workflow_id="1">
   <work_order_id>8464</work_order_id>
   <name>create file</name>
   <workflow id="1" revision_id="14">create file</workflow>
   <status state="Created">WorkOrder created at Tue Nov 25 07:24:01 CST
2014</status>
   <comments/>
   <parameters></parameters>
   <ownership>
      <initiated_by>admin</initiated_by>
      <running_as>system</running_as>
      <launched_by>admin</launched_by>
   </ownership>
   <priority dedicated_worker="false"/>
   <work_steps></work_steps>
</work_order>
```

A 400 Bad Request is returned if a mandatory input parameter is absent from the initiate work order request.

If a HTTP POST method is issued, the body can be any user defined XML data.

In this case, the workflow must expect the XML payload with a run-time input defined as "Raw_post_body" and the first step must be the one parsing the data such as:



Usage for an asynchronous request with JSON response:

http://<Orchestrator_IP_address>/aspera/orchestrator/api/initiate/<workflow_id>.json?login=admin&password=aspera&external_parameters[<name_of run-time input1>]=<value of run-time input1>&external_parameters[<name_of run-time input2>]=<value of run-time input2>…

Example:

http://localhost/aspera/orchestrator/api/initiate/1.json?login=admin&password=aspera&external_parameters[foo]=bar

Sample returned JSON:


```
{
   "work_order":    {
      "workflow_revision_id": 14,
      "name": "create file",
      "initiate_date": null,
      "forkedAt": null,
      "derivedFrom": null,
      "workflowName": "create file",
      "created_at": "2014-11-25T13:25:18Z",
      "cleanup_after_days": null,
      "monitor_id": null,
      "comments": null,
      "updated_at": "2014-11-25T13:25:18Z",
      "priority": null,
```

```
      "max_running": 0,
      "master_id": 8465,
      "launched_by": 1,
      "initiatedBy": 1,
      "branchedFrom": null,
      "statusDetails": "WorkOrder created at Tue Nov 25 07:25:18 CST
2014",
      "purge_after_days": null,
      "id": 8465,
      "workflow_id": 1,
      "branchedFromOrder": null,
      "higherPriority": null,
      "completion_date": null,
      "status": "Created",
      "running_as": 2
   },
   "parameters": [],
   "work_steps": []
}
```


Usage for a synchronous request with XML response:

This call will create a work order and block until the specified step in the workflow has been executed. A variable name can be specified to get an output variable value. Only the variable value comes in output (no XML wrapper around).

http://<Orchestrator_IP_address>/aspera/orchestrator/api/initiate/xml?login=admin&password=aspera&work_order[workflow_id]=<workflow id>&external_parameters[<name_of run-time input1>]=<value of run-time input1>&external_parameters[<name_of run-time input-n>]=<value of run-time input-n>&synchronous=true&explicit_output_step=<step_name>&explicit_output_variable=<variable_name>

Example:

http://localhost/aspera/orchestrator/api/initiate/xml?login=admin&password=aspera&work_order[workflow_id]=1&external_parameters[foo]=bar&synchronous=true&explicit_output_step=create xml content&explicit_output_variable=Generated_file_content

Sample returned value:

"hello"

CHECK THE STATUS OF A SPECIFIC WORK ORDER

Description:

A method to poll the status of a specific work order, identified by its work order ID. The result includes both the overall status for the work-order as well as the individual status for all its composing steps.

If the work order includes sub-workflows, only the sub-workflow status is displayed, not the status for the steps within the sub-workflow.

The response can be either in XML or JSON format.

Usage with XML result:

http://<Orchestrator_IP_address>/aspera/orchestrator/api/work_order_status/<work order_id>?login=admin&password=aspera

Example:

http://localhost/aspera/orchestrator/api/work_order_status/3109?login=admin&password=aspera

The required input parameter is:

- workflow_reporter/work_order_status/<work order_id> : specifies the work order id

Usage for a request with JSON response including all steps:

http://<Orchestrator_IP_address>/aspera/orchestrator/api/work_order_status/<work order_id>.json?login=admin&password=aspera

Example:

http://localhost/aspera/orchestrator/api/work_order_status/344.json?login=admin&password=aspera

Usage for a request with XML response including all steps:

```
<?xml version="1.0"?>
<work_order time="2014-11-25 13:11:35 UTC" action="status" id="8461"
master_id="8461">
   <name>tf1 demo duplicate</name>
   <workflow id="256" revision_id="5">tf1 demo duplicate</workflow>
   <status state="In Progress">Executing Step: user input
(46755)</status>
   <comments/>
   <latest_update/>
   <ownership>
      <initiated_by>admin</initiated_by>
      <running_as>system</running_as>
      <launched_by>admin</launched_by>
   </ownership>
   <priority dedicated_worker="false">2</priority>
   <timestamps>
      <created_at>2014-11-24 16:48:37 UTC</created_at>
      <initiated_at>2014-11-24 16:48:38 UTC</initiated_at>
```

```
      <updated_at>2014-11-24 16:48:40 UTC</updated_at>
      <completed_at/>
   </timestamps>
   <work_steps>
      <work_step id="46756">
         <name>Create faspex package</name>
         <status state="In Progress" attempt="1">FASPEX Authorization
received</status>
         <action type="FaspexDelivery" id="2"/>
         <created_at>2014-11-24 16:48:37 UTC</created_at>
         <initiated_at>2014-11-24 16:48:39 UTC</initiated_at>
         <updated_at>2014-11-24 16:48:39 UTC</updated_at>
         <completed_at>2014-11-24 16:48:39 UTC</completed_at>
      </work_step>
      …
   </work_steps>
</work_order>
```

Sample returned JSON:

```
{
   "work_order":    {
      "workflow_revision_id": 5,
      "name": "tf1 demo duplicate",
      "initiate_date": "2014-11-24T16:48:38Z",
      "forkedAt": null,
      "derivedFrom": null,
      "workflowName": "tf1 demo duplicate",
      "created_at": "2014-11-24T16:48:37Z",
      "cleanup_after_days": null,
      "monitor_id": null,
      "comments": "",
      "updated_at": "2014-11-24T16:48:40Z",
      "priority": 2,
      "max_running": 0,
      "master_id": 8461,
      "launched_by": 1,
      "initiatedBy": 1,
      "branchedFrom": null,
      "statusDetails": "Executing Step: user input (46755)",
      "purge_after_days": null,
      "id": 8461,
      "workflow_id": 256,
      "branchedFromOrder": null,
      "higherPriority": null,
      "completion_date": null,
      "status": "In Progress",
      "running_as": 2
   },
   "work_steps":    [
            {
         "workOrder_id": 8461,
         "preProcessing": null,
         "group": 8461,
         "executionTimeout": 0,
```

```
            "activable_date": "2014-11-24T16:48:39Z",
            "stepName": null,
            "created_at": "2014-11-24T16:48:37Z",
            "updated_at": "2014-11-24T16:48:39Z",
            "timeout": null,
            "on_timeout": null,
            "step_type": "FaspexDelivery",
            "attempt": 1,
            "step_id": 3810,
            "statusDetails": "FASPEX Authorization received",
            "rank": 1,
            "id": 46756,
            "activation_date": "2014-11-24T16:48:39Z",
            "workflow_id": 256,
            "synch_filter": 0,
            "completion_date": "2014-11-24T16:48:39Z",
            "action_id": 2,
            "status": "In Progress",
            "running_as": 2,
            "original_activation_date": "2014-11-24T16:48:39Z",
            "workStepName": "Create faspex package",
            "postProcessing": null
        },
                …
    ]
}
```

CHECK THE STATUS OF A SPECIFIC STEP

A method to poll the status of a specific step, identified by its work order ID and step name.

If the work order includes sub-workflows, only the sub-workflow status is displayed, not the status for the steps within the sub-workflow.

The response can be either in XML or JSON format.

Usage for a request with XML response:

http://<Orchestrator_IP_address>/aspera/orchestrator/api/work_step_status/work_order.xml?login=admin&password=aspera&work_order_id=<work order_id>&step_name=<step_name>

Example:

http://localhost/aspera/orchestrator/api/work_step_status/work_order.xml?login=admin&password=aspera&work_order_id=8461&step_name=Create faspex package

Sample returned XML:

```
<?xml version="1.0"?>
<Work-step-status>
   <inputs type="array"/>
   <carry-thrus type="array">
      <carry-thru>
         <name>myfiles</name>
         <id type="integer">60067</id>
         <type>string</type>
         <value>/root/chris/data_for_demos/shiporder.cmd</value>
      </carry-thru>
   </carry-thrus>
   <work-step>
      <workOrder-id type="integer">8461</workOrder-id>
      <preProcessing nil="true"/>
      <group type="integer">8461</group>
      <executionTimeout type="integer">0</executionTimeout>
      <activable-date type="datetime">2014-11-24T16:48:39Z</activable-
date>
      <stepName nil="true"/>
      <created-at type="datetime">2014-11-24T16:48:37Z</created-at>
      <updated-at type="datetime">2014-11-24T16:48:39Z</updated-at>
      <timeout nil="true"/>
      <on-timeout nil="true"/>
      <step-type>FaspexDelivery</step-type>
      <attempt type="integer">1</attempt>
      <step-id type="integer">3810</step-id>
      <statusDetails>FASPEX Authorization received</statusDetails>
      <rank type="integer">1</rank>
      <id type="integer">46756</id>
      <activation-date type="datetime">2014-11-
24T16:48:39Z</activation-date>
      <workflow-id type="integer">256</workflow-id>
      <synch-filter type="integer">0</synch-filter>
```

```
      <completion-date type="datetime">2014-11-
24T16:48:39Z</completion-date>
      <action-id type="integer">2</action-id>
      <status>Complete</status>
      <running-as type="integer">2</running-as>
      <original-activation-date type="datetime">2014-11-
24T16:48:39Z</original-activation-date>
      <workStepName>Create faspex package</workStepName>
      <postProcessing nil="true"/>
   </work-step>
   <outputs type="array">
      <output>
         <name>package_id</name>
         <id type="integer">60070</id>
         <type>string</type>
         <value>5e0f9336-5e58-46a5-b944-a8253a956041</value>
      </output>
      <output>
         <name>package_post_url</name>
         <id type="integer">60071</id>
         <type>string</type>
<value><![CDATA[fasp://faspex@testchris2.sl.dev.asperacloud.net:33001/T
ranscoded - 5e0f9336-5e58-46a5-b944-a8253a956041.aspera-package/PKG -
Transcoded?token=ATM3_ACsvFz3yt1byrRDQ7cEtFKg0wxWS6ubjsWnkA8IAxSlbKoAAE
WHQkLFXfeLjDz5Eq8uDH7_3MTA&cookie=aspera.faspex20:u:90ce20d7]]></value>
      </output>
   </outputs>
</Work-step-status>
```

Usage for a request with JSON response:

http://<Orchestrator_IP_address>/aspera/orchestrator/api/work_step_status/work_order.json?login=admin&password=aspera&work_order_id=<work order_id>&step_name=<step_name>

Example:

http://localhost/aspera/orchestrator/api/work_step_status/work_order.json?login=admin&password=aspera&work_order_id=8461&step_name=Create faspex package

Sample returned JSON:

```
{
   "inputs": [],
   "carry_thrus": [   {
      "name": "myfiles",
      "id": 60067,
      "type": "string",
      "value": "/root/chris/data_for_demos/shiporder.cmd"
   }],
   "work_step":    {
      "workOrder_id": 8461,
      "preProcessing": null,
      "group": 8461,
      "executionTimeout": 0,
      "activable_date": "2014-11-24T16:48:39Z",
      "stepName": null,
      "created_at": "2014-11-24T16:48:37Z",
```

```
      "updated_at": "2014-11-24T16:48:39Z",
      "timeout": null,
      "on_timeout": null,
      "step_type": "FaspexDelivery",
      "attempt": 1,
      "step_id": 3810,
      "statusDetails": "FASPEX Authorization received",
      "rank": 1,
      "id": 46756,
      "activation_date": "2014-11-24T16:48:39Z",
      "workflow_id": 256,
      "synch_filter": 0,
      "completion_date": "2014-11-24T16:48:39Z",
      "action_id": 2,
      "status": "Complete",
      "running_as": 2,
      "original_activation_date": "2014-11-24T16:48:39Z",
      "workStepName": "Create faspex package",
      "postProcessing": null
   },
   "outputs":    [
            {
         "name": "package_id",
         "id": 60070,
         "type": "string",
         "value": "5e0f9336-5e58-46a5-b944-a8253a956041"
      },
            {
         "name": "package_post_url",
         "id": 60071,
         "type": "string",
         "value":
"fasp://faspex@testchris2.sl.dev.asperacloud.net:33001/Transcoded -
5e0f9336-5e58-46a5-b944-a8253a956041.aspera-package/PKG -
Transcoded?token=ATM3_ACsvFz3yt1byrRDQ7cEtFKg0wxWS6ubjsWnkA8IAxSlbKoAAE
WHQkLFXfeLjDz5Eq8uDH7_3MTA&cookie=aspera.faspex20:u:90ce20d7-17c3-4672"
      }
   ]
}
```


You can obtain only the step status from the step name with:

http://<Orchestrator_IP_address>/aspera/orchestrator/api/work_order.xml?login=admin&password=aspera&work_order_id=<work_order_id>&step_name=<step_name>&status_only=true

Example:

https://localhost/aspera/orchestrator/api/work_step_status/work_order.xml?login=admin&password=aspera&work_order_id=8461&step_name=Create faspex package&status_only=true

Sample returned value:

```
Complete
```

You can obtain only the step status from the step id with:

http://<Orchestrator_IP_address>/aspera/orchestrator/api/work_step_status/<step_id>.txt?login=admin&password=aspera&status_only=true

Example:

https://localhost/aspera/orchestrator/aspera/orchestrator/api/work_step_status/46756.txt?login=admin&password=aspera&status_only=true

Sample returned value:

```
Complete
```

An output value expressed can be obtained from the step id. This id can be obtained with the previous call (e.g. <id type="integer">46756</id> for the XML response).

Usage for a request with text response:

http://<Orchestrator_IP_address>/aspera/orchestrator/api/work_step_status/<step_id>.txt?login=admin&password=aspera&output_name=<output_name>

Example:

http://localhost/aspera/orchestrator/api/work_step_status/46756.txt?login=admin&password=aspera&output_name=package_id

Sample returned value:

```
5e0f9336-5e58-46a5-b944-a8253a956041
```

For a complex type, the output value can be expressed in XML with:

Usage for a request with XML response:

http://localhost/aspera/orchestrator/api/work_step_status/252158.xml?login=admin&password=aspera&output_name=IndexedTable

Sample returned value:

```
<?xml version="1.0"?>
<Work_step_output>
   <name>IndexedTable</name>
   <id type="integer">301979</id>
   <value>{'Transcode server' => {"node_id"=>"transcoder",
"node"=>"Transcode server", "capacity_kb"=>"30000"}, 'Mac' =>
{"node_id"=>"localhost", "node"=>"Mac",
"capacity_kb"=>"20000"}}</value>
   <type>hash</type>
</Work_step_output>
```

Usage for a request with JSON response:

http://localhost/aspera/orchestrator/api/work_step_status/252158.json?login=admin&password=aspera&output_name=IndexedTable

Sample returned value:


```
{"Mac":{"capacity_kb":"20000","node":"Mac","node_id":"localhost"},"Transcode server":{"capacity_kb":"30000","node":"Transcode server","node_id":"transcoder"}}
```


CANCEL A WORK ORDER

Description:

An API method to cancel a specific work order, identified by its work order ID. The result includes both the overall status for the work-order as well as the individual status for all its composing steps.

Adding the force option will ensure that the work-order is cancelled even if it is not responding to polite cancel request to cancel itself. If the force flag is not provided, the work order status is Failed.

Note: if the work-order was already complete (or failed), then the returned status will indicate the actual status.

Usage:

http://<Orchestrator_IP_address>/aspera/orchestrator/api/work_order_cancel/<work_order_id>?login=admin&password=aspera

Example:

http://localhost/aspera/orchestrator/api/work_order_cancel/3109?login=admin&password=aspera

or

http://localhost/aspera/orchestrator/api/work_order_cancel/3109?login=admin&password=aspera&force=true

The required input parameter is:

- workflow_reporter/work_order_cancel/<work_order_id> : specifies the work order id

Sample returned XML

```
<?xml version="1.0"?>
<work_order time="2012-03-02 03:45:05 UTC" action="cancel" id="3109"
master_id="5309">
  <name>test after-end</name>
  <workflow id="22" revision_id="7">test after-end</workflow>
  <status state="Canceled">Canceled at Thu Mar 01 19:45:05 -0800
2012</status>
  <comments></comments>
  <ownership>
    <initiated_by>admin</initiated_by>
    <running_as>admin</running_as>
    <launched_by>admin</launched_by>
  </ownership>
  <priority dedicated_worker="false">2</priority>
  <timestamps>
    <created_at>2012-03-02 03:44:48 UTC</created_at>
    <initiated_at>2012-03-02 03:44:50 UTC</initiated_at>
    <updated_at>2012-03-02 03:45:05 UTC</updated_at>
    <completed_at>2012-03-02 03:45:05 UTC</completed_at>
  </timestamps>
  <work_steps>
        <work_step id="37601">
          <name>never</name>
          <status state="Inactive" attempt="0"></status>
          <action type="MergePoint" id="1"/>
          <created_at>2012-03-02 03:44:48 UTC</created_at>
          <initiated_at></initiated_at>
          <updated_at>2012-03-02 03:44:48 UTC</updated_at>
          <completed_at></completed_at>
        </work_step>
        <work_step id="37602">
          <name>gnarp</name>
          <status state="Obsolete" attempt="1">Work Order Processing is
finished</status>
          <action type="CustomRuby" id="12"/>
          <created_at>2012-03-02 03:44:48 UTC</created_at>
          <initiated_at>2012-03-02 03:44:50 UTC</initiated_at>
          <updated_at>2012-03-02 03:45:05 UTC</updated_at>
          <completed_at></completed_at>
        </work_step>
  </work_steps>
</work_order>
```

CANCEL A WORK STEP

Description:

An API method to cancel a specific work step, identified by its work step ID. The response lists the work step cancelled and the current status along with timestamps.

Usage:

http://<Orchestrator_IP_address>/aspera/orchestrator/api/work_step_cancel/<work_order_id>?login=admin&password=aspera

Example:

http://localhost/aspera/orchestrator/api/work_step_cancel/3109?login=admin&password=aspera

The required input parameter is:

- workflow_reporter/work_step_cancel/<work_order_id> : specifies the work step id

Sample returned XML

```
<?xml version="1.0"?>
  <work_step time="2015-05-18 04:53:20 UTC" action="cancel" id="" >
    <name>Test input</name>
    <workflow id="553"></workflow>
    <status state="Canceling">Awaiting user input at Sun May 17
21:53:17 PDT 2015</status>
    <action type="UserInput" id="20"/>
    <status state="Canceling" attempt="1">Awaiting user input at Sun
May 17 21:53:17 PDT 2015</status>
    <timestamps>
     <created_at>2015-05-13 15:22:38 UTC</created_at>
      <initiated_at>2015-05-13 15:22:41 UTC</initiated_at>
      <updated_at>2015-05-18 04:53:20 UTC</updated_at>
      <completed_at></completed_at>
    </timestamps>
  </work_step>
```

RESET A WORK ORDER

Description:

This is a method to reset a specific work order, identified by its work order ID. The result includes both the overall status for the work-order as well as the individual status for all its composing steps.

Usage:

http://<Orchestrator_IP_address>/aspera/orchestrator/api/work_order_reset/<work_order_id>?login=admin&password=aspera

Example:

http://localhost/aspera/orchestrator/api/work_order_reset/3109?login=admin&password=aspera

The required input parameter is:

- workflow_reporter/work_order_reset/<work_order_id> : specifies the work step id

Sample returned XML

```
<?xml version="1.0"?>
<work_order time="2015-05-18 04:55:18 UTC" action="status" id="1227"
master_id="1227">
  <name>Demo - user task</name>
  <workflow id="553" revision_id="3">Demo - user task</workflow>
  <status state="Created">WorkOrder reset at Sun May 17 21:55:18 PDT
2015</status>
  <comments/>
  <latest_update>reset</latest_update>
  <ownership>
    <initiated_by>admin</initiated_by>
    <running_as>system</running_as>
    <launched_by>admin</launched_by>
  </ownership>
  <priority dedicated_worker="false">2</priority>
  <timestamps>
    <created_at>2015-05-13 15:22:38 UTC</created_at>
    <initiated_at>2015-05-13 15:22:39 UTC</initiated_at>
    <updated_at>2015-05-18 04:55:18 UTC</updated_at>
    <completed_at>2015-05-18 04:53:21 UTC</completed_at>
  </timestamps>
    <work_steps>
          <work_step id="8269" category="System">
            <name>Logic prior</name>
            <status state="Inactive" attempt="0">reset</status>
            <action type="MergePoint" id="69"/>
            <created_at>2015-05-13 15:22:38 UTC</created_at>
            <initiated_at>2015-05-13 15:22:40 UTC</initiated_at>
            <updated_at>2015-05-18 04:55:18 UTC</updated_at>
            <completed_at>2015-05-13 15:22:40 UTC</completed_at>
          </work_step>
          <work_step id="8270" category="System">
            <name>Logic prior 1</name>
            <status state="Inactive" attempt="0">reset</status>
```

```
    <action type="MergePoint" id="70"/>
    <created_at>2015-05-13 15:22:38 UTC</created_at>
    <initiated_at>2015-05-13 15:22:40 UTC</initiated_at>
    <updated_at>2015-05-18 04:55:18 UTC</updated_at>
    <completed_at>2015-05-13 15:22:40 UTC</completed_at>
  </work_step>
 <work_step id="8271" category="System">
    <name>Post User input</name>
    <status state="Inactive" attempt="0">reset</status>
    <action type="MergePoint" id="71"/>
    <created_at>2015-05-13 15:22:38 UTC</created_at>
    <initiated_at/>
    <updated_at>2015-05-18 04:55:18 UTC</updated_at>
    <completed_at/>
  </work_step>
 <work_step id="8272" category="User Interactions">
    <name>Test input</name>
    <status state="Inactive" attempt="0">reset</status>
    <action type="UserInput" id="20"/>
    <created_at>2015-05-13 15:22:38 UTC</created_at>
    <initiated_at>2015-05-13 15:22:41 UTC</initiated_at>
    <updated_at>2015-05-18 04:55:18 UTC</updated_at>
    <completed_at>2015-05-18 04:53:21 UTC</completed_at>
  </work_step>
 <work_step id="8273">
    <name>Workflow Start</name>
    <status state="False" attempt="">reset</status>
    <action type="WorkOrderEnd" id=""/>
    <created_at>2015-05-13 15:22:38 UTC</created_at>
    <initiated_at/>
    <updated_at>2015-05-18 04:55:18 UTC</updated_at>
    <completed_at/>
  </work_step>
 <work_step id="8274">
    <name>Workflow Failed</name>
    <status state="False" attempt="">reset</status>
    <action type="WorkOrderEnd" id=""/>
    <created_at>2015-05-13 15:22:38 UTC</created_at>
    <initiated_at/>
    <updated_at>2015-05-18 04:55:18 UTC</updated_at>
    <completed_at/>
  </work_step>
  <work_step id="8275">
    <name>Workflow End</name>
    <status state="False" attempt="">reset</status>
    <action type="WorkOrderEnd" id=""/>
    <created_at>2015-05-13 15:22:38 UTC</created_at>
    <initiated_at/>
    <updated_at>2015-05-18 04:55:18 UTC</updated_at>
    <completed_at/>
  </work_step>
 <work_step id="8276">
    <name>Error Encountered</name>
    <status state="False" attempt="">reset</status>
    <action type="WorkOrderEnd" id=""/>
    <created_at>2015-05-13 15:22:38 UTC</created_at>
    <initiated_at/>
```

```
            <updated_at>2015-05-18 04:55:18 UTC</updated_at>
            <completed_at/>
          </work_step>
    </work_steps>
</work_order>
```

FETCH OUTPUT SPECIFICATION OF A WORKFLOW

Description:

This is a method to fetch the output specification of a workflow (e.g. step names and step variable names without the variable values).

The inputs to this API method are a workflow id and authentication parameters (login and password) if passing them inline. The user must have privileges to view workflow details. The output of this API call is an XML that contains the details such as variable name, type, step name etc.

Usage:

http://<Orchestrator_IP_address>/aspera/orchestrator/api/workflow_outputs_spec?login=admin&password=aspera&workflow_id=<workflow_id>

Example:

http://localhost/aspera/orchestrator/api/workflow_outputs_spec?login=admin&password=aspera&workflow_id=878

The required input parameter is:

- workflow_id=<workflow_id>: specifies the workflow id

Sample returned XML

```
<?xml version="1.0"?>
<workflow_outputs_spec time="2012-09-14 23:04:50 UTC" action="opspec">
      <context id="5513">
        <workflow_id>878</workflow_id>
        <step_name>Parameter</step_name>
        <variable_name>input</variable_name>
        <value_type>string</value_type>
        <variable_type></variable_type>
        <created_at>2012-09-11 00:31:22 UTC</created_at>
        <updated_at>2012-09-11 00:31:22 UTC</updated_at>
      </context>
      <context id="5512">
        <workflow_id>878</workflow_id>
<step_name>Test FedWF</step_name>
        <variable_name>message</variable_name>
        <value_type>string</value_type>
        <variable_type></variable_type>
        <created_at>2012-09-11 00:31:22 UTC</created_at>
        <updated_at>2012-09-11 00:31:22 UTC</updated_at>
      </context>
</work_order_outputs_spec>
```

FETCH OUTPUT OF A WORK ORDER

Description:

This is a method to fetch the output of a work order (e.g. step names and step variable names with the variable values).

XML or JSON data representation can be returned in the response.

The inputs to this API method are a work order id, step name, variable name whose output is desired and authentication parameters (login and password) if passing them inline. The user must have privileges to view work order details.

The output of the API call is an XML structure that contains the details such as variable name, type, value, step name etc. It is possible that a workflow contains the same variable name in different steps. Use the step name input to drill down to the step you desire to fetch the output variable from.

Another URL form allows to get a variable value rather than the long-form XML wrapper around the variable.

Spaces are accepted in the URL for variable names (e.g for step_name). Surrounding single or double quotes are not accepted.

Usage for a request with XML long-form response:

http://<Orchestrator_IP_address>/aspera/orchestrator/api/work_order_output?login=admin&password=aspera&work order_id=<work order_id>&step_name=<step_name>&variable_name=<variable_name>

Alternatively, this URL can be used:

http://<Orchestrator_IP_address>/ aspera/orchestrator/api/work_order_output/<work order_id>.xml?login=admin&password=aspera&step_name=<step_name >&variable_name=<variable_name>

The required input parameters are

- work order_id=<work order_id>: specifies the work order id
- step_name=<step_name>: specifies the step name
- variable_name=<variable_name>: specifies the variable name within a step

Example 1: Find an output for a given step and a given variable

http://localhost/aspera/orchestrator/api/work_order_output?work order_id=11081&step_name=Test2&variable_name=Count_of_ContentFiles&login=admin&password=aspera

Sample returned XML:

```
<?xml version="1.0"?>
<work_order_output time="2014-03-14 04:50:56 UTC" action="output">
   <variable id="29039" name="Test2:Count_of_ContentFiles">
      <work order_id>11081</work order_id>
      <step_name>Test2</step_name>
      <variable_name>Count_of_ContentFiles</variable_name>
      <value_type>int</value_type>
      <value>3</value>
      <variable_type>OUTPUT</variable_type>
   </variable>
</work_order_output>
```

Example 2: Find output(s) for a given variable

http://localhost/aspera/orchestrator/api/work_order_output?work order_id=1106&variable_name=FileName&login=admin&password=aspera

Sample returned XML:

```
<?xml version="1.0"?>
<work_order_output time="2014-03-14 02:19:41 UTC" action="output">
   <variable id="28785" name="Update failure:FileName">
      <work order_id>11068</work order_id>
      <step_name>Update failure</step_name>
      <variable_name>FileName</variable_name>
      <value_type>string</value_type>
      <value>FileName</value>
      <variable_type>OUTPUT</variable_type>
   </variable>
   <variable id="28788" name="Update success:FileName">
      <work order_id>11068</work order_id>
      <step_name>Update success</step_name>
      <variable_name>FileName</variable_name>
      <value_type>string</value_type>
      <value/>
      <variable_type>OUTPUT</variable_type>
   </variable>
   <variable id="28789" name="Scan for incoming files:FileName">
      <work order_id>11068</work order_id>
      <step_name>Scan for incoming files</step_name>
      <variable_name>FileName</variable_name>
      <value_type>string</value_type>
      <value>/demo/comcast/to_disney/wall_e_5.avi</value>
      <variable_type>OUTPUT</variable_type>
   </variable>
</work_order_output>
```

Example 3: Find all output(s) for a given work order

http://localhost/aspera/orchestrator/api/work_order_output?work order_id=1106&login=admin&password=aspera

Sample returned XML:

```
<?xml version="1.0"?>
<work_order_output time="2014-03-24 21:25:47 UTC" action="output">
   <variable id="29629" name="Successful ingest
notification:FullMessage">
      <work order_id>11109</work order_id>
      <step_name>Successful ingest notification</step_name>
     <variable_name>FullMessage</variable_name>
      <value_type>string</value_type>
<value>&lt;MessageDefinition&gt;&lt;Destinations&gt;&lt;To&gt;pavan+sup
port@asperasoft.com&lt;/To&gt;&lt;/Destinations&gt;&lt;Subject&gt;Inges
t successful of wall_e_12.avi for
Turner&lt;/Subject&gt;&lt;/MessageDefinition&gt;
</value>
      <variable_type>OUTPUT</variable_type>
   </variable>
   <variable id="29630" name="Ingest failure:FullMessage">
      <work order_id>11109</work order_id>
      <step_name>Ingest failure</step_name>
      <variable_name>FullMessage</variable_name>
      <value_type>string</value_type>
      <value></value>
      <variable_type>OUTPUT</variable_type>
   </variable>
   <variable id="29631" name="Update failure:manifest">
      <work order_id>11109</work order_id>
      <step_name>Update failure</step_name>
      <variable_name>manifest</variable_name>
      <value_type>string</value_type>
      <value></value>
      <variable_type>OUTPUT</variable_type>
   </variable>
   <variable id="29632" name="Update failure:manifest_id">
      <work order_id>11109</work order_id>
      <step_name>Update failure</step_name>
      <variable_name>manifest_id</variable_name>
      <value_type>int</value_type>
      <value></value>
      <variable_type>OUTPUT</variable_type>
   </variable>
   <variable id="29637" name="Scan for incoming files:FileName">
      <work order_id>11109</work order_id>
      <step_name>Scan for incoming files</step_name>
      <variable_name>FileName</variable_name>
      <value_type>string</value_type>
      <value>/demo/comcast/to_turner/wall_e_12.avi</value>
      <variable_type>OUTPUT</variable_type>
   </variable>
</work_order_output>
```

Example 4: Find all output(s) in a workflow for a single step

http://localhost/aspera/orchestrator/api/work_order_output?work order_id=1106&login=admin&password=aspera&step_name=test

Sample returned XML:

```
<?xml version="1.0"?>
<work_order_output time="2014-03-24 19:07:15 UTC" action="output">
   <variable id="29638" name="Forward:Transferred_File_list">
      <work order_id>11109</work order_id>
      <step_name>Forward</step_name>
      <variable_name>Transferred_File_list</variable_name>
      <value_type>array</value_type>
      <value>["/demo/comcast/to_turner/wall_e_12.avi"]</value>
      <variable_type>OUTPUT</variable_type>
   </variable>
   <variable id="29639" name="Forward:Transfer_Stats">
      <work order_id>11109</work order_id>
      <step_name>Forward</step_name>
      <variable_name>Transfer_Stats</variable_name>
      <value_type>string</value_type>
      <value>{:FilesFailed=>0, :JobRetryCount=>0,
:FilesTransferring=>0, :BytesWritten=>4830720,
:BytesTransferred=>4830720, :sessionId=>["918b2456-095c-4ce5-9b33-
af8551f4deb9"], :FilesComplete=>1}</value>
      <variable_type>OUTPUT</variable_type>
   </variable>
   <variable id="29640" name="Forward:Job_ID">
      <work order_id>11109</work order_id>
      <step_name>Forward</step_name>
      <variable_name>Job_ID</variable_name>
      <value_type>string</value_type>
      <value>f698d602-d6e6-45a4-b82a-df5f65621a2e</value>
      <variable_type>OUTPUT</variable_type>
   </variable>
   <variable id="29641" name="Forward:Transferred_Files_md5">
      <work order_id>11109</work order_id>
      <step_name>Forward</step_name>
      <variable_name>Transferred_Files_md5</variable_name>
      <value_type>hash</value_type>
      <value/>
      <variable_type>OUTPUT</variable_type>
   </variable>
</work_order_output>
```

Usage for a request with value only response:

http://<Orchestrator_IP_address>/api/work_order_output/<work order_id>.string?login=admin&password=aspera&step_name=<step_name>&variable_name=<variable_name>

Example:

http://localhost/aspera/orchestrator/api/work_order_output/11081.string?step_name=Test2&variable_name=Count_of_ContentFiles&login=admin&password=aspera

Sample returned data:

```
3
```

Error use case: a step name or variable name is incorrect

This is returned:

```
<work_order_output time="2014-11-05 08:43:36 UTC" action="output"></work_order_output>
```

Error use case: The step has not yet been executed when the API request is made

An XML structure without a value is returned such as:

```
<?xml version="1.0"?>
<work_order_output time="2014-03-14 04:50:56 UTC" action="output">
   <variable id="29039" name="Test2:Count_of_ContentFiles">
      <work order_id>11081</work order_id>
      <step_name>Test2</step_name>
      <variable_name>Count_of_ContentFiles</variable_name>
      <value_type>int</value_type>
      <value/>
      <variable_type>OUTPUT</variable_type>
   </variable>
</work_order_output>
```

Usage for a request with JSON response:

http://<Orchestrator_IP_address>/api/work_order_output/<work order_id>.json?login=admin&password=aspera&step_name=<step_name>&variable_name=<variable_name>

The JSON response holds only the value for the specified step+variable.

Example:

http://localhost/aspera/orchestrator/api/work_order_output/11081.json?step_name=Test2&variable_name=Count_of_ContentFiles&login=admin&password=aspera

Sample returned JSON:

```
"3"
```

LIST WORKORDERS OF A WORKFLOW

Description:

This is a method for external applications to fetch work orders associated with a workflow. Currently, 3 conditions can be given with this method:

1. From date
2. To date
3. Maximum results
4. Status
5. Priority

Usage for a request with XML response:

http://localhost/aspera/orchestrator/api/work_orders_list/<workflow_id>.xml?login=admin&password=aspera&from_date=<from_date>&to_date=<to_date>&max_results=<maximum_results>

Example:
http://localhost/aspera/orchestrator/api/work_orders_list/8738.json?login=admin&password=aspera&from_date='06-27-2015 02:40:59'&to_date='06-29-2015 02:40:59'&max_results=100

The required input parameters are

- Workflow_id : specifies the workflow from which the work orders would be listed

The optional input parameters are

- from_date=<from date> : specifies the 'from date' condition
- to_date=<to_date> : specifies the 'to date' condition
- max_results=<maximum_results> : specifies the 'maximum results' condition

Sample returned XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<work-orders type="array">
  <work-order>
    <branchedFrom type="integer" nil="true"></branchedFrom>
    <branchedFromOrder type="integer" nil="true"></branchedFromOrder>
    <cleanup-after-days type="integer" nil="true"></cleanup-after-days>
    <comments nil="true"></comments>
    <completion-date type="datetime">2015-11-06T19:03:28Z</completion-
date>
    <created-at type="datetime">2015-11-06T19:03:27Z</created-at>
    <derivedFrom type="integer" nil="true"></derivedFrom>
    <forkedAt type="integer" nil="true"></forkedAt>
    <higherPriority type="boolean" nil="true"></higherPriority>
    <id type="integer">491</id>
    <initiate-date type="datetime">2015-11-06T19:03:28Z</initiate-date>
    <initiatedBy type="integer">1</initiatedBy>
```

```
    <label nil="true"></label>
    <launched-by type="integer">1</launched-by>
    <master-id type="integer">491</master-id>
    <max-running type="integer">0</max-running>
    <monitor-id type="integer" nil="true"></monitor-id>
    <name>test</name>
    <priority type="integer" nil="true"></priority>
    <purge-after-days type="integer" nil="true"></purge-after-days>
    <running-as type="integer">2</running-as>
    <status>Complete</status>
    <statusDetails>WorkOrder ended at Fri Nov 06 13:03:28 CST 2015 with
status: In Progress</statusDetails>
    <tags></tags>
    <updated-at type="datetime">2015-11-06T19:03:28Z</updated-at>
    <workflowName>test</workflowName>
    <workflow-id type="integer">1</workflow-id>
    <workflow-revision-id type="integer">24</workflow-revision-id>
  </work-order>
  <work-order>
    <branchedFrom type="integer" nil="true"></branchedFrom>
    <branchedFromOrder type="integer" nil="true"></branchedFromOrder>
    <cleanup-after-days type="integer" nil="true"></cleanup-after-days>
    <comments nil="true"></comments>
    <completion-date type="datetime">2015-11-06T19:46:12Z</completion-
date>
    <created-at type="datetime">2015-11-06T19:46:10Z</created-at>
    <derivedFrom type="integer" nil="true"></derivedFrom>
    <forkedAt type="integer" nil="true"></forkedAt>
    <higherPriority type="boolean" nil="true"></higherPriority>
    <id type="integer">492</id>
    <initiate-date type="datetime">2015-11-06T19:46:11Z</initiate-date>
    <initiatedBy type="integer">1</initiatedBy>
    <label nil="true"></label>
    <launched-by type="integer">1</launched-by>
    <master-id type="integer">492</master-id>
    <max-running type="integer">0</max-running>
    <monitor-id type="integer" nil="true"></monitor-id>
    <name>test</name>
    <priority type="integer" nil="true"></priority>
    <purge-after-days type="integer" nil="true"></purge-after-days>
    <running-as type="integer">2</running-as>
    <status>Complete</status>
    <statusDetails>WorkOrder ended at Fri Nov 06 13:46:12 CST 2015 with
status: In Progress</statusDetails>
    <tags></tags>
    <updated-at type="datetime">2015-11-06T19:46:12Z</updated-at>
    <workflowName>test</workflowName>
    <workflow-id type="integer">1</workflow-id>
    <workflow-revision-id type="integer">24</workflow-revision-id>
  </work-order>
</work-orders type="array">
```

Usage for a request with JSON response:

http://localhost/aspera/orchestrator/api/work_orders_list/<workflow_id>.json?login=admin&password=aspera&from_date=<from_date>&to_date=<to_date>&max_results=<maximum_results>

Sample returned JSON:


```
 { "work_orders":[
      {
         "forkedAt":null,
         "workflow_revision_id":24,
         "tags":"||",
         "created_at":"2015-11-06T19:03:27Z",
         "workflow_id":1,
         "updated_at":"2015-11-06T19:03:28Z",
         "derivedFrom":null,
         "comments":null,
         "cleanup_after_days":null,
         "statusDetails":"WorkOrder ended at Fri Nov 06 13:03:28 CST 20
15 with status: In Progress",
         "master_id":491,
         "label":null,
         "initiate_date":"2015-11-06T19:03:28Z",
         "higherPriority":null,
         "status":"Complete",
         "purge_after_days":null,
         "monitor_id":null,
         "max_running":0,
         "branchedFromOrder":null,
         "workflowName":"test",
         "launched_by":1,
         "branchedFrom":null,
         "name":"test",
         "id":491,
         "running_as":2,
         "completion_date":"2015-11-06T19:03:28Z",
         "priority":null,
         "initiatedBy":1
      },
      {
         "forkedAt":null,
         "workflow_revision_id":24,
         "tags":"||",
         "created_at":"2015-11-06T19:46:10Z",
         "workflow_id":1,
         "updated_at":"2015-11-06T19:46:12Z",
         "derivedFrom":null,
         "comments":null,
         "cleanup_after_days":null,
         "statusDetails":"WorkOrder ended at Fri Nov 06 13:46:12 CST 20
15 with status: In Progress",
         "master_id":492,
         "label":null,
         "initiate_date":"2015-11-06T19:46:11Z",
         "higherPriority":null,
```

```
        "status":"Complete",
        "purge_after_days":null,
        "monitor_id":null,
        "max_running":0,
        "branchedFromOrder":null,
        "workflowName":"test",
        "launched_by":1,
        "branchedFrom":null,
        "name":"test",
        "id":492,
        "running_as":2,
        "completion_date":"2015-11-06T19:46:12Z",
        "priority":null,
        "initiatedBy":1
      }
   ]
}
```


APPLICATION STATUS AND MANAGEMENT

ORCHESTRATOR BACKGROUND PROCESSES STATUS

Description:

A method to poll the status of Orchestrator processes. If available, the PID for each process is returned.

Usage:

http://localhost/aspera/orchestrator/api/processes_status/0?login=admin&password=aspera

Sample returned XML:


```
<?xml version="1.0"?>
  <processes time="2012-01-25 08:23:10 UTC" action="status" id="0">
  <process class="engine" pid="9825" status="running"/>
  <process class="mongrel" pid="1337" status="running" port="3000"/>
  <process class="mongrel" pid="1340" status="running" port="3001"/>
  <process class="mongrel" pid="1343" status="running" port="3002"/>
  <process class="monitor" pid="96279" status="running"/>
  <process class="worker" id="0" pid="13752" status="running"
type="asynchronous"/>
  <process class="worker" id="1" pid="9863" status="running"
type="normal"/>
  <process class="worker" id="2" pid="9865" status="running"
type="normal"/>
</processes>
```

CONTROL PROCESS

Description:

This is a method for external applications to control orchestrator processes:

1. Engine
2. Monitor
3. Mongrel
4. Worker

Usage for Engine and Monitor

http://localhost/aspera/orchestrator/api/control_process/<process_name>.xml?login=admin&password=pass&command=<command>

Example:

http://localhost/aspera/orchestrator/api/control_process/engine.xml?login=admin&password=pass&command='kill'

The commands for each process:

1. Engine – kill , start, stop
2. Monitor – kill, start, stop
3. Mongrel – kill, start, unlock_port
4. Worker – kill, start, stop

Usage for Mongrel

http://localhost/aspera/orchestrator/api/control_process/<process_name>.xml?login=admin&password=pass&command=<command>&port=<port>

http://localhost:3000/aspera/orchestrator/api/control_process/mongrel.json?login=admin&password=admin&command=kill&port=3002

Usage for Worker

http://localhost/aspera/orchestrator/api/control_process/<process_name>.xml?login=admin&password=pass&command=<command>&worker_id=<worker_id>

Sample returned XML for Engine Start:

```
<processes time="2015-12-03 10:29:19
UTC" action="status" engine_id="0">
     <process class="engine" pid="83296" status="running"/>
     <process class="mongrel" pid="83301" status="running" port="3000"/>
     <process class="mongrel" pid="83303" status="running" port="3001"/>
     <process class="mongrel" pid="83305" status="running" port="3002"/>
     <process class="monitor" pid="83299" status="running"/>
     <process class="worker" id="0" pid="83312" status="running" type="a
     synchronous"/>
     <process class="worker" id="1" pid="83314" status="running" type="a
     synchronous"/>
```

```
    <process class="worker" id="2" pid="83316" status="running" type="n
    ormal"/>
    <process class="worker" id="3" pid="83318" status="running" type="n
    ormal"/>
    <process class="worker" id="4" pid="83320" status="running" type="n
    ormal"/>
    <process class="worker" id="5" pid="83310" status="running" type="n
    ormal"/>
</processes>
```

Sample returned JSON for Mongrel Kill on port 3002:


```
{
   "processes":{
      "process":[
         {
            "pid":25907,
            "class":"engine",
            "status":"running"
         },
         {
            "pid":25912,
            "port":3000,
            "class":"mongrel",
            "status":"running"
         },
         {
            "pid":25914,
            "port":3001,
            "class":"mongrel",
            "status":"running"
         },
         {
            "pid":"-",
            "port":3002,
            "class":"mongrel",
            "status":"not running"
         },
         {
            "pid":25910,
            "class":"monitor",
            "status":"running"
         },
         {
            "id":"0",
            "type":"asynchronous",
            "pid":25923,
            "class":"worker",
            "status":"running"
         },
         {
            "id":"1",
            "type":"asynchronous",
            "pid":25925,
            "class":"worker",
            "status":"running"
         },
         {
            "id":"2",
            "type":"normal",
```

```
            "pid":25927,
            "class":"worker",
            "status":"running"
         },
         {
            "id":"3",
            "type":"normal",
            "pid":25929,
            "class":"worker",
            "status":"running"
         },
         {
            "id":"4",
            "type":"normal",
            "pid":25931,
            "class":"worker",
            "status":"running"
         },
         {
            "id":"5",
            "type":"normal",
            "pid":25921,
            "class":"worker",
            "status":"running"
         }
      ],
      "engine_id":0,
      "action":"status",
      "time":"2016-01-12 23:34:03"
   }
}
```

ORCHESTRATOR MONITOR

Description:

A method to poll the status of the workflow, folder and script monitors.

Usage:

http://localhost/aspera/orchestrator/api/monitor_snapshot/all?login=admin&password=aspera

Return the status for all the monitors.

More restrictive view can be obtained using:

http://localhost/aspera/orchestrator/api/monitor_snapshot/workflows?login=admin&password=aspera
http://localhost/aspera/orchestrator/api/monitor_snapshot/folders?login=admin&password=aspera
http://localhost:/aspera/orchestrator/api/monitor_snapshot/scripts?login=admin&password=aspera

Sample returned XML:

```
<?xml version="1.0"?>
<monitor time="2012-05-10 05:47:53 UTC" action="snapshot" id="all">
    <monitor_workflows>
        <monitor_workflow id="1" workflow_id="1" workflow_name="ENG
Contributions" active="true" >
          <status>No Issues</status>
            <status_report>In Progress=>1</status_report>
            <last_activity>2012-05-09 23:40:15</last_activity>
            <last_polled>2012-05-10 05:47:51 UTC</last_polled>
        </monitor_workflow>
        <monitor_workflow id="2" workflow_id="5" workflow_name="Non ENG
Contributions" active="false" >
          <status>Monitoring Off</status>
        </monitor_workflow>
        <monitor_workflow id="3" workflow_id="14" workflow_name="Livex
OnDemand distribution" active="false" >
          <status>Monitoring Off</status>
        </monitor_workflow>
    </monitor_workflows>
    <monitor_folders>
        <monitor_folder id="1" folder="temporary folder" path="/tmp"
active="false">
          <status>Monitoring Off</status>
        </monitor_folder>
        <monitor_folder id="2" folder="Other tmp" path="/tmp"
active="true">
          <status>Alerts</status>
            <status_report>Aging entries (> 2 min)</status_report>
            <content>7 files, 1 folders</content>
```

```
            <age_range>3353 - 36065 min.</age_range>
            <last_polled>2012-05-10 05:46:59 UTC</last_polled>
        </monitor_folder>
    </monitor_folders>
    <monitor_scripts>
        <monitor_script id="1" script="warn" node="localhost"
path="/opt/aspera/scripts/warn.sh"  active="false">
          <status>Monitoring Off</status>
        </monitor_script>
        <monitor_script id="2" script="chill" node="localhost"
path="/opt/aspera/scripts/ok.sh"  active="true">
          <status>No Issues</status>
            <status_report>All is good</status_report>
            <last_polled>2012-05-10 05:47:52 UTC</last_polled>
        </monitor_script>
    </monitor_scripts>
</monitor>
```

PING A REMOTE ORCHESTRATOR INSTANCE

Description:

This is a method to fetch the Orchestrator version, operating system information and licensing information.
This method may be used in a Federated workflow to check a remote Orchestrator status before remote/delegated requests are sent to it.

Usage:

http://<orchestrator-IP-address>/aspera/orchestrator/api/remote_node_ping/

Sample returned XML:

```
<?xml version="1.0"?>
<remote_orchestrator_info>
     <orchestrator-version>2.1.0.92065-00008</orchestrator-version>
     <platform>x86_64-pc-linux-gnu</platform>
     <engine_id>0</engine_id>
     <license-info>
          <licensee>AsperaColo</licensee>
          <can_execute>true</can_execute>
          <max_workflows>0</max_workflows>
          <comments>Orchestrator Server</comments>
          <max_processes>0</max_processes>
          <can_design>true</can_design>
          <mac-address>any</mac-address>
     </license-info>
</remote_orchestrator_info>
```

Sample returned JSON:


```
{
   "remote_orchestrator_info":{
      "orchestrator-version":"2.3.5.xxxxx-00003",
      "license-info":{
         "mac-address":"any, any",
         "licensee":"development, development",
         "max_workflows":0,
         "can_execute":true,
         "can_design":true,
         "max_processes":0,
         "comments":"Permanent dev license -
 generated by Marc, Permanent dev license - generated by Marc"
      },
      "engine_id":0,
      "platform":"i686-apple-darwin14.1.0"
   }
}
```


Note: This method does not require any authentication parameters.

GENERAL

PERSIST CUSTOM DATA

Description:

This is a method for external applications to submit data to Orchestrator that is persisted in a database table (shared_states). Example scenario – Rhozet WFS can send a Web notification upon transcode complete or failure. This API method requires authentication.

To simplify searching for the data entered in the table, users can pass two fields to make each entry "uniquely identifiable". The fields are 'identifier' which is a string and 'uniq_id' which is an integer.

All key-value pairs (except for login, password, identifier, uniq_id) will be stored into a serialized hash in the shared_states table.

The database entries can be searched in two ways:

MySQL query:

select id, entry from shared_states where aggregate_type = "identifier" and aggregate_id = uniq_id

Ruby query:

SharedState.find(:all, :conditions => ["aggregate_type = ? and aggregate_id = ?", identifier, uniq_id])

Usage:

http://localhost/aspera/orchestrator/api/custom_api_call?login=admin&password=aspera&identifier=<>&job_id=123&<key_1>=<value_1>...&<key_n>=<value_n>

The required input parameters are (either one of them or both of them can be specified):

- identifier=<identifier>: specifies a custom data identifier (type string)
- uniq_id=<another identifier>: specifies a custom data unique identifier (type integer)

The optional input parameters are:

- <data_name_1>=<data_1_value>: specifies some data to pass in
- …
- <data_name_n>=<data_n_value>: specifies some data to pass in

Example:

http://localhost/aspera/orchestrator/api/custom_api_call?login=admin&password=aspera&identifier=testing&uniq_id=321&job_id=123&job_result=success

Sample returned XML:

```
<?xml version="1.0"?>
<api_call action="custom_api_call">
   <shared_state>
      <id>34</id>
      <creation_time>"2014-03-22 21:11:35 UTC"</creation_time>
   </shared_state>
</api_call>
```

DASHBOARD

Description:

This is a method for external applications to access Orchestrator dashboard.

Any URL used to access the dashboard can be called externally with the inline login/password.

Example for the first page in the dashboard:

http://localhost/aspera/orchestrator/frames/home/0?login=admin&password=aspera

To enable inline credentials for the dashboard, this parameter must be set in orchestrator.yml:

inline_auth_allowed: true

This also applies to direct start pages (to manually start a workflow via a customized HTML page from an external application):

http://localhost/aspera/orchestrator/work_orders/direct_start/1?login=admin&password=aspera

QUEUES

FETCH QUEUED ITEMS FROM A QUEUE

Description:

This is a method for external applications to fetch list of queued items from an Orchestrator queue.

Usage for a request with XML response:

http://<Orchestrator_IP>/aspera/orchestrator/api/queued_items/<queue_name>.<format>?login=admin&password=aspera

The required input parameters are

- queue_name=< queue_name >: specifies the queue name

Example:

http://localhost/aspera/orchestrator/api/queued_items/transcode.xml?login=admin&password=aspera

Sample returned XML:

```
<?xml version="1.0"?>
<Queue>
  <Queued_items>
    <Queued_item>
      <priority>100</priority>
      <weight>3859</weight>
      <queued_item>/demo/to_transcoder/3_10_to_Yuma.avi</queued_item>
      <id>587</id>
      <updated_at>Tue Jan 06 21:38:51 UTC 2015</updated_at>
      <position>50</position>
      <originator_type></originator_type>
      <originator_id>64623</originator_id>
      <item_description>3_10_to_Yuma</item_description>
      <rank>0</rank>
      <created_at>Tue Jan 06 19:29:01 UTC 2015</created_at>
    </Queued_item>
    <Queued_item>
      <priority>100</priority>
      <weight>12823</weight>
      <queued_item>/demo/to_transcoder/39steps_train.avi</queued_item>
      <id>586</id>
      <updated_at>Tue Jan 06 19:28:54 UTC 2015</updated_at>
      <position>100</position>
      <originator_type></originator_type>
      <originator_id>64621</originator_id>
      <item_description>39steps_train</item_description>
      <rank>1</rank>
      <created_at>Tue Jan 06 19:28:54 UTC 2015</created_at>
    </Queued_item>
  </Queued_items>
  <Queue_metadata>
    <comments>ad_hoc creation for QueueStager #4</comments>
    <administrators>
      <group>1</group>
      <user>3</user>
    </administrators>
    <id>transcode</id>
    <paused>true</paused>
    <name>transcode</name>
 </Queue_metadata>
</Queue>
```

Usage for a request with JSON response:

http://localhost/aspera/orchestrator/api/queued_items/transcode.json?login=admin&password=aspera

Sample returned JSON:

```
{
    "queue": {
        "name": "transcode",
        "comments": "ad_hoc creation for QueueStager #4",
```

```
        "administrators": {
            "groups": [
                1
            ],
            "users": [
                3
            ]
        },
        "paused": true,
        "id": "transcode"
    },
    "queued_items": [
        {
            "item_description": "3_10_to_Yuma",
            "weight": 3859,
            "position": 50,
            "originator_id": 64623,
            "originator_type": null,
            "updated_at": "2015-01-06T21:38:51Z",
            "rank": 0,
            "queued_item": "/demo/to_transcoder/3_10_to_Yuma.avi",
            "priority": 100,
            "id": 587,
            "created_at": "2015-01-06T19:29:01Z"
        },
        {
            "item_description": "39steps_train",
            "weight": 12823,
            "position": 100,
            "originator_id": 64621,
            "originator_type": null,
            "updated_at": "2015-01-06T19:28:54Z",
            "rank": 1,
            "queued_item": "/demo/to_transcoder/39steps_train.avi",
```

```
        "priority": 100,
        "id": 586,
        "created_at": "2015-01-06T19:28:54Z"
    }
]}
```

LOOKUP QUEUED ITEM FROM ORCHESTRATOR QUEUE

Description:

This API method provides the ability to search for a queued item in one or more Orchestrator Queues

Usage

1. Search for queued item in a specific queue. The queued item provided in the URL must be a string match to the queued item in the database

For XML response:

http://localhost/aspera/orchestrator/api/lookup_queued_item?queue_name=CP_ENT_FFMPEG&format=xml&queued_item=test&login=admin&password=aspera

Sample returned XML:

```
<Queue>
    <Queue_metadata>
        <comments>ad_hoc creation for QueueStager #37</comments>
        <administrators></administrators>
        <name>CP_ENT_FFMPEG</name>
        <paused>false</paused>
    </Queue_metadata>
    <Queued_items>
        <Queued_item>
            <updated_at>Wed Apr 29 22:06:50 UTC 2015</updated_at>
            <originator_type/>
            <id>26</id>
            <created_at>Wed Apr 29 22:06:50 UTC 2015</created_at>
            <queued_item>test</queued_item>
            <position>100</position>
            <rank>0</rank>
            <weight>0</weight>
            <originator_id>1217</originator_id>
            <item_description>test</item_description>
            <priority>100</priority>
        </Queued_item>
    </Queued_items>
</Queue>
```

For JSON response:

http://localhost/aspera/orchestrator/api/lookup_queued_item?queue_name=CP_ENT_FFMPEG&format=json&queued_item=test&login=admin&password=aspera

Sample returned JSON:

```
{
   "queue":{
      "comments":"ad_hoc creation for QueueStager #21",
      "administrators":{
         "users":[],
         "groups":[
```

```
            1,
            2,
            4,
            5,
            6
         ]
      },
      "name":"Dev_QC_Action",
      "paused":false
   },
   "queued_items":[
      {
         "updated_at":"2015-12-04T08:22:40Z",
         "originator_type":null,
         "id":28,
         "created_at":"2015-12-04T08:22:40Z",
         "queued_item":"test_add",
         "position":100,
         "rank":0,
         "weight":0,
         "originator_id":null,
         "item_description":"",
         "priority":100
      }
   ]
}
```


2. Search by a queue item id. The value of the queued item id can be obtained from the Fetch queued items API method described in page 36

For XML response:

http://localhost/aspera/orchestrator/api/lookup_queued_item?format=xml&queued_item_id=29&login=admin&password=aspera

Sample returned XML:


```
<Queue>
    <Queue_metadata>
        <comments>ad_hoc creation for QueueStager #21</comments>
        <administrators>
            <group>1</group>
            <group>2</group>
            <group>4</group>
            <group>5</group>
            <group>6</group>
        </administrators>
        <name>Dev_QC_Action</name>
        <paused>false</paused>
    </Queue_metadata>
    <Queued_items>
        <Queued_item>
            <originator_type/>
            <created_at>Fri Dec 04 08:23:49 UTC 2015</created_at>
            <position>110</position>
            <rank>1</rank>
```

```
            <updated_at>Fri Dec 04 08:23:49 UTC 2015</updated_at>
            <originator_id/>
            <priority>100</priority>
            <weight>0</weight>
            <queued_item>test_add1</queued_item>
            <item_description/>
            <id>29</id>
        </Queued_item>
    </Queued_items>
</Queue>
```

For JSON response:

http://localhost/aspera/orchestrator/api/lookup_queued_item?format=json&queued_item_id=29&login=admin&password=aspera

Sample returned JSON:

```
{
   "queued_items":[
      {
         "originator_type":null,
         "created_at":"2015-12-04T08:23:49Z",
         "position":110,
         "rank":1,
         "updated_at":"2015-12-04T08:23:49Z",
         "originator_id":null,
         "priority":100,
         "weight":0,
         "queued_item":"test_add1",
         "item_description":"",
         "id":29
      }
   ],
   "queue":{
      "comments":"ad_hoc creation for QueueStager #21",
      "administrators":{
         "users":[],
         "groups":[
            1,
            2,
            4,
            5,
            6
         ]
      },
      "name":"Dev_QC_Action",
      "paused":false
   }
}
```

REORDER QUEUED ITEM

Description:

This API method provides the ability to reorder a queued item. Four reorder operations are supported – up (move up by one rank), down (move down by one rank), first (move to the top of the queue), last (move to the bottom of the queue). In the API responses, only the rank field changes if an operation was successfully performed.

Rank 0 indicates the queued item is on top of the queue.

Usage

1. Search for queued item in a specific queue. The queued item provided in the URL must be a string match to the queued item in the database

For XML response:

http://localhost/aspera/orchestrator/api/reorder_queued_item?queue_name=CP_ENT_FFMPEG&format=xml&queued_item=test&login=admin&password=aspera&operation=up

Sample returned XML:

```
<Queue>
    <Queue_metadata>
        <comments>ad_hoc creation for QueueStager #37</comments>
        <administrators></administrators>
        <name>CP_ENT_FFMPEG</name>
        <paused>false</paused>
    </Queue_metadata>
    <Queued_items>
        <Queued_item>
            <updated_at>Wed Apr 29 22:06:50 UTC 2015</updated_at>
            <originator_type/>
            <id>26</id>
            <created_at>Wed Apr 29 22:06:50 UTC 2015</created_at>
            <queued_item>test</queued_item>
            <position>100</position>
            <rank>0</rank>
            <weight>0</weight>
            <originator_id>1217</originator_id>
            <item_description>test</item_description>
            <priority>100</priority>
        </Queued_item>
    </Queued_items>
</Queue>
```

For JSON response:

http://localhost/aspera/orchestrator/api/reorder_queued_item?queue_name=CP_ENT_FFMPEG&format=json&queued_item=test&login=admin&password=aspera&operation=first

Sample returned JSON:


```
{
   "queue":{
      "comments":"ad_hoc creation for QueueStager #21",
      "administrators":{
         "users":[],
         "groups":[
            1,
            2,
            4,
            5,
            6
         ]
      },
      "name":"Dev_QC_Action",
      "paused":false
   },
   "queued_items":[
      {
         "updated_at":"2015-12-04T08:22:40Z",
         "originator_type":null,
         "id":28,
         "created_at":"2015-12-04T08:22:40Z",
         "queued_item":"test_add",
         "position":100,
         "rank":0,
         "weight":0,
         "originator_id":null,
         "item_description":"",
         "priority":100
      }
   ]
}
```


2. Search by a queue item id. The value of the queued item id can be obtained from the Fetch queued items API method described in page 36 or the Lookup queued item API method in page 54

For XML response:

http://localhost/aspera/orchestrator/api/reorder_queued_item?format=xml&queued_item_id=29&login=admin&password=aspera&operation=down

Sample returned XML:


```
<Queue>
    <Queue_metadata>
        <comments>ad_hoc creation for QueueStager #21</comments>
        <administrators>
            <group>1</group>
            <group>2</group>
            <group>4</group>
            <group>5</group>
            <group>6</group>
        </administrators>
```

```
        <name>Dev_QC_Action</name>
        <paused>false</paused>
    </Queue_metadata>
    <Queued_items>
        <Queued_item>
            <originator_type/>
            <created_at>Fri Dec 04 08:23:49 UTC 2015</created_at>
            <position>110</position>
            <rank>1</rank>
            <updated_at>Fri Dec 04 08:23:49 UTC 2015</updated_at>
            <originator_id/>
            <priority>100</priority>
            <weight>0</weight>
            <queued_item>test_add1</queued_item>
            <item_description/>
            <id>29</id>
        </Queued_item>
    </Queued_items>
</Queue>
```

For JSON response:

http://localhost/aspera/orchestrator/api/reorder_queued_item?format=json&queued_item_id=29&login=admin&password=aspera&operation=down

Sample returned JSON:

```
{
   "queued_items":[
      {
         "originator_type":null,
         "created_at":"2015-12-04T08:23:49Z",
         "position":110,
         "rank":1,
         "updated_at":"2015-12-04T08:23:49Z",
         "originator_id":null,
         "priority":100,
         "weight":0,
         "queued_item":"test_add1",
         "item_description":"",
         "id":29
      }
   ],
   "queue":{
      "comments":"ad_hoc creation for QueueStager #21",
      "administrators":{
         "users":[],
         "groups":[
            1,
            2,
            4,
            5,
            6
         ]
      },
      "name":"Dev_QC_Action",
      "paused":false
```

```
    }
}
```

TASKS

LIST TASKS OF A USER

Description:

This is a method for external applications to list the tasks associated with a user.

Usage for a request with XML response:

http://localhost/aspera/orchestrator/api/list_tasks/<user>.xml?login=admin&password=admin

Example:

http://localhost/aspera/orchestrator/api/list_tasks/admin.xml?login=admin&password=admin

The required input parameters are

User: specifies the user whose tasks are to be listed.

Sample returned XML:

```
<User_tasks>
  <task>
    <event>Create a new Endpoint</event>
    <requested_user_inputs>
      <value_type>hash</value_type>
      <name>Step_information</name>
      <required>false</required>
    </requested_user_inputs>
    <requested_user_inputs>
      <value_type>string</value_type>
      <name>Emails</name>
      <required>true</required>
    </requested_user_inputs>
    <requested_user_inputs>
      <value_type>string</value_type>
      <name>Login</name>
      <required>true</required>
    </requested_user_inputs>
    <requested_user_inputs>
      <value_type>string</value_type>
      <name>Name</name>
      <required>true</required>
    </requested_user_inputs>
    <userInput_id>2</userInput_id>
    <state_id>2626</state_id>
    <created_at>Fri Dec 04 03:38:23 UTC 2015</created_at>
    <updated_at>Fri Dec 04 03:38:23 UTC 2015</updated_at>
    <role_id>1</role_id>
    <status>Assigned</status>
    <completedBy></completedBy>
    <group>Administrator</group>
    <user_id></user_id>
    <id>1</id>
```

```
</task>
<task>
  <event>Create a new Endpoint</event>
  <requested_user_inputs>
    <value_type>hash</value_type>
    <name>Step_information</name>
    <required>false</required>
  </requested_user_inputs>
  <requested_user_inputs>
    <value_type>string</value_type>
    <name>Emails</name>
    <required>true</required>
  </requested_user_inputs>
  <requested_user_inputs>
    <value_type>string</value_type>
    <name>Login</name>
    <required>true</required>
  </requested_user_inputs>
  <requested_user_inputs>
    <value_type>string</value_type>
    <name>Name</name>
    <required>true</required>
  </requested_user_inputs>
  <userInput_id>2</userInput_id>
  <state_id>2626</state_id>
  <created_at>Fri Dec 04 03:38:23 UTC 2015</created_at>
  <updated_at>Fri Dec 04 03:38:23 UTC 2015</updated_at>
  <role_id></role_id>
  <status>Assigned</status>
  <completedBy></completedBy>
  <user>admin</user>
  <user_id>1</user_id>
  <id>2</id>
</task>
```

```
  <user>admin</user>
</User_tasks>
```

Usage for a request with JSON response:

http://localhost/aspera/orchestrator/api/list_tasks/<user>.xml?login=admin&password=admin

Sample returned JSON :


```
{
   "tasks":[
      {
         "event":"Create a new Endpoint",
         "requested_user_inputs":[
            {
               "value_type":"hash",
               "name":"Step_information",
               "required":false
            },
            {
               "value_type":"string",
               "name":"Emails",
               "required":true
            },
            {
               "value_type":"string",
               "name":"Login",
               "required":true
            },
            {
               "value_type":"string",
               "name":"Name",
               "required":true
            }
         ],
         "userInput_id":2,
         "state_id":2626,
         "created_at":"2015-12-04T03:38:23Z",
         "updated_at":"2015-12-04T03:38:23Z",
         "role_id":1,
         "status":"Assigned",
         "completedBy":null,
         "group":"Administrator",
         "user_id":null,
         "id":1
      },
      {
         "event":"Create a new Endpoint",
         "requested_user_inputs":[
            {
               "value_type":"hash",
               "name":"Step_information",
               "required":false
            },
            {
               "value_type":"string",
               "name":"Emails",
```

```
                "required":true
            },
            {
                "value_type":"string",
                "name":"Login",
                "required":true
            },
            {
                "value_type":"string",
                "name":"Name",
                "required":true
            }
        ],
        "userInput_id":2,
        "state_id":2626,
        "created_at":"2015-12-04T03:38:23Z",
        "updated_at":"2015-12-04T03:38:23Z",
        "role_id":null,
        "status":"Assigned",
        "completedBy":null,
        "user":"admin",
        "user_id":1,
        "id":2
      }
   ],
   "user":"admin"
}
```

FETCH DETAILS OF A TASK

Description:

This is a method for external applications to fetch the details of a task. You can identify by:

1. Task ID
2. Workorder ID

1. IDENTIFY BY TASK ID

http://localhost/aspera/orchestrator/api/task_details/<task_id>.<format>?login=admin&password=admin

Example:

Xml:

http://localhost/aspera/orchestrator/api/task_details /1.xml?login=admin&password=admin

JSON:

http://localhost/aspera/orchestrator/api/task_details /1.json?login=admin&password=admin

Sample returned XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<User_task>
   <task_inputs />
   <task>
      <userInput_id>2</userInput_id>
      <state_id>2626</state_id>
      <created_at>Fri Dec 04 03:38:23 UTC 2015</created_at>
      <updated_at>Fri Dec 04 03:38:23 UTC 2015</updated_at>
      <role_id>1</role_id>
      <status>Assigned</status>
      <completedBy />
      <user_id />
      <id>1</id>
   </task>
   <user_inputs>
      <type>hash</type>
      <value />
      <name>Step_information</name>
```

```
      <required>false</required>
   </user_inputs>
   <user_inputs>
      <type>string</type>
      <value />
      <name>Emails</name>
      <required>true</required>
   </user_inputs>
   <user_inputs>
      <type>string</type>
      <value />
      <name>Login</name>
      <required>true</required>
   </user_inputs>
   <user_inputs>
      <type>string</type>
      <value />
      <name>Name</name>
      <required>true</required>
   </user_inputs>
</User_task>
```

Sample returned JSON:


```
{
   "task_inputs":{

   },
   "task":{
      "active_assignment":{
         "userInput_id":2,
         "state_id":2626,
         "created_at":"2015-12-04T03:38:23Z",
         "updated_at":"2015-12-04T03:38:23Z",
         "role_id":1,
         "status":"Assigned",
         "completedBy":null,
         "user_id":null,
         "id":1
      }
   },
   "user_inputs":[
      {
         "type":"hash",
         "value":null,
         "name":"Step_information",
         "required":false
      },
      {
         "type":"string",
         "value":null,
         "name":"Emails",
         "required":true
      },
      {
         "type":"string",
         "value":null,
         "name":"Login",
         "required":true
      },
      {
         "type":"string",
         "value":null,
         "name":"Name",
         "required":true
      }
   ]
}
```

2. IDENTIFY BY WORKORDER ID

When identifying a Workorder in a Workflow with a single task:

http://localhost/aspera/orchestrator/api/task_details/WO<work order_id>.<format>?login=admin&password=admin

When identifying a work order in a workflow with multiple tasks:

http://localhost/aspera/orchestrator/api/task_details/WO<work order_id>.<format>?login=admin&password=admin&step_name=<step_name>

Example:
http://localhost/aspera/orchestrator/api/task_details/WO518.xml?login=admin&password=admin &step_name= Request+approval

SUBMIT TASK

Description:

This is a method for external applications to submit a task. You can submit as:

1. User
2. Admin on behalf of a user

1. USER

http://localhost/aspera/orchestrator/api/submit_task/<task_id>.<format>?login=admin&password=pass&outputs[Filename]=<output_param>

Xml:

http://localhost/aspera/orchestrator/api/submit_task/1.xml?login=admin&password=pass&outputs[Filename]=tron.avi

JSON:

http://localhost/aspera/orchestrator/api/submit_task/1.json?login=admin&password=pass&outputs[Filename]=tron.avi

Sample returned XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<User_task>
   <input_provider>admin</input_provider>
   <task>
      <userInput_id>2</userInput_id>
      <state_id>2626</state_id>
      <created_at>Fri Dec 04 03:38:23 UTC 2015</created_at>
      <updated_at>Fri Dec 04 06:01:55 UTC 2015</updated_at>
      <role_id>1</role_id>
      <status>Complete</status>
      <completedBy>1</completedBy>
      <user_id />
      <id>1</id>
   </task>
   <user_inputs>Filenametron.avi</user_inputs>
</User_task>
```

Sample returned JSON:


```
{
   "input_provider":"admin",
   "task":{
      "active_assignment":{
         "userInput_id":2,
         "state_id":2626,
         "created_at":"2015-12-04T03:38:23Z",
         "updated_at":"2015-12-04T06:01:55Z",
         "role_id":1,
         "status":"Complete",
         "completedBy":1,
         "user_id":null,
         "id":1
      }
   },
   "user_inputs":{
      "Filename":"tron.avi",
    }
}
```


2. ADMIN ON BEHALF OF A USER

Admin can submit a task on behalf of a user by passing the 'input_provider' parameter:

http://localhost/aspera/orchestrator/api/submit_task/<task_id>.<format>?login=admin&password=admin &input_provider=<user>&outputs[Filename]=<output_param>

Example:

http://localhost/aspera/orchestrator/api/submit_task/55.xml?login=admin&password=pass&input_provider='lola'&outputs[Filename]=tron.avi

Sample returned:


```
{
   "input_provider":"lola",
   "task":{
      "active_assignment":{
         "userInput_id":2,
         "state_id":2626,
         "created_at":"2015-12-04T03:38:23Z",
         "updated_at":"2015-12-04T06:01:55Z",
         "role_id":1,
         "status":"Complete",
         "completedBy":1,
         "user_id":null,
         "id":1
      }
   },
   "user_inputs":{
      "Filename":"tron.avi"
    }
}
```

INLINE VALIDATION WITH ORCHESTRATOR

INTRODUCTION

Inline validation is a concept within ASCP to invoke external REST end points during the life cycle of a transfer session. There REST calls to end points can occur at:

a) SESSION_START

b) SESSION_STOP

c) SESSION_RUNNING, the moment the threshold is met (optional event).

Threshold value can be configured via the **validation_threshold_kb** tag.

More details about Inline File Validation using Aspera Transfer products can be obtained here: https://developer.asperasoft.com/web/api/inline-validation/index

ASSUMPTIONS

This section assumes that

- Aspera Orchestrator v2.0 or newer has been installed with standard options
- Aspera Transfer product(s) (Faspex, Connect, Enterprise server etc.) have been installed with standard options
- ASCP binary v 3.5
- Port 80 and 443 are open between the Aspera Transfer server and Orchestrator.

STEP 1: WORKFLOW

a) Design and publish a workflow in Orchestrator (or)

b) Import a workflow and publish it

Mandatory:

Make sure the workflow contains a runtime parameter with the name “File” (without double quotes) and as a type “Hash”

The File parameter is populated by a Hash similar to the below (by Orchestrator) when ASCP triggers an API call

```
{'size' => "1048576000", 'end_byte' => "1048576000", 'file_csum_type'
=> "none", 'session_id' => "ab658c6e-648f-4249-abe9-d5c3801ef534",
'target_rate_kbps' => "10000", 'cipher' => "aes-128",
'file_name_encoding' => "utf16", 'min_rate_kbps' => "0", 'startstop' =>
"running", 'user_id' => "0", 'client_ip' => "50.19.236.72", 'host' =>
"66.211.109.190", 'rate_policy' => "fair", 'file' =>
"/LargeFiles/Washington U", 'user' => "aspera-nyc", 'direction' =>
"recv", 'start_byte' => "0"}
```

The workflow can use the “file” key for processing. The “startstop” key can be used to route your workflow.

STEP 2: ASPERA URL

Follow instructions in Admin guide to obtain the Aspera call URL of the workflow published in Step 1. The URL should comprise of Hostname, route to Orchestrator controller and the workflow id. Make sure to add the port to the URL if different from the default port (443)

STEP 3: ASPERA CONFIGURATION

Locate the aspera.conf file typically at /opt/aspera/etc/aspera.conf (on a Unix host)

Choose a transfer user that will be used for inline validation. In this example, the user “faspex” will be used. Add the text highlighted in green to the file. Note that in between the validate_uri tags the URL obtained in Step 2 must be inserted.

```
<user>
        <session_timeout_sec>60</session_timeout_sec>
        <name>faspex</name>
        <transfer>
                <validation_uri>
https://localhost/aspera/orchestrator/external_calls/validate/35?user=admin&pass
word=admin
                </validation_uri>
                <validation_threshold_kb>1000</validation_threshold_kb>
        </transfer>
</user>
```

The text in blue can be inserted if the validation takes more than 30 seconds to avoid timeout. The value between session_timeout_sec tags can be increased to an appropriate number based on the environment.

STEP 4: ASCP FILE AND ASPERA CONNECT

Check the ASCP binary version by execution “ascp –A” command. If not 3.5 or above, please contact Aspera Support for upgrade checks.

Aspera Connect browser plugin must be v.3.4 or higher for the error messages to correctly propagate to the user.

STEP 5: MONGREL CONFIGURATION

For Production servers, it is recommended to separate Mongrel usage for ASCP calls and user usage of Orchestrator. Follow instructions in admin guide to increase the number of mongrels. Restart of Apache is required.

ERROR CODES IN ORCHESTRATOR

Currently the error messages delivered by Aspera Orchestrator are text based, and rely on clients parse the error message to decipher the issue. The HTTP error code associated with the response is generally HTTP 200 Ok along with a response body.

Error message	HTTP Error Code
XML Format: *<?xml version="1.0"?>* *<error time="<--Timestamp -->" action="<-- action -->" id="<-- id -->">* *<-- error_message -->* *</error>*	200
JSON Format: {error:{“action”:”#{action}”,”time”:”#{Timestamp}”,”id”:”#{id}”,”description”:#{error_message }}}"	200
When a bad request is sent i.e. with invalid data, clients will see this error code returned in the response	400
When authentication fails with the provided user/password or user/api_key or user/basicauth or user/scrambled password, clients will see this error code returned in the response	401
When authorization fails with the provided user/password or user/api_key or user/basicauth or user/scrambled password, clients will see this error code returned in the response	403
When a Orchestrator Mongrel (UI rendering daemon) is terminated due to the longevity of the request, clients will see a Proxy error	502
When Orchestrator Mongrels (UI rendering daemon) are not running, clients will see this error code returned in the response	503
When the Apache services is stopped, users will see connection refused errors on their clients	-